Preface

Natural Language Processing (NLP) has benefited from promising recent advances including the employment of latest deep learning technology amongst a host of other solutions. The current pandemic has prevented the in-person exchange of ideas and networking of NLP researchers and students, but virtual communication opportunities have enabled continued collaboration and provided alternative communication channels. While eagerly awaiting the return of normality, the international conference RANLP’2021 is offering an opportunity to promote the latest advances in the field, exchange ideas, learn from each other and interact virtually.

The conference in 2021 features nine keynote speakers:

- Tim Baldwin (The University of Melbourne, Australia),
- Josef van Genabith and Nico Herbig (DFKI and Saarland University, Germany),
- He He (New York University, USA),
- Eduard Hovy (Carnegie Mellon University, USA),
- Jing Jiang (Singapore Management University, Singapore),
- Alessandro Moschitti (Amazon Alexa, USA),
- Hwee Tou Ng (National University of Singapore, Singapore),
- Constantin Orasan (University of Surrey, UK),
- Sebastian Riedel (University College London and Facebook AI Research, UK).

This year 257 papers were submitted to the event. The good news is that most of them were of very good quality: 50 regular papers, 66 short papers, 68 posters, and 2 demos were accepted for presentation at the conference. The higher acceptance rate translates into better research quality in 2021 (with numerous papers presented in parallel made possible due to the online mode of communication).

The proceedings cover a wide variety of NLP topics, including but not limited to: deep learning; machine translation; opinion mining and sentiment analysis; author profiling, offensive language detection, fact checking; semantics and discourse; topic modelling; named entity recognition; coreference resolution; word-sense disambiguation; knowledge discovery and language models; corpora and ontologies; syntax; text summarisation; text categorisation, information extraction, question answering; multilinguality; language tools; NLP for biomedical domain; and NLP for low-resource languages.

In 2021 RANLP will host two post-conference workshops on popular NLP topics: the 14th Workshop on Building and Using Comparable Corpora (BUCC) and the First Workshop on Multimodal Machine Translation for Low Resource Languages (MMTLLRL-2021). The First International CLaDA-BG Conference, to be held on 6-7 September 2021, will present the achievements of the Bulgarian Interdisciplinary Research e-Infrastructure CLADA-BG for resources and technologies for the Bulgarian linguistic and cultural heritage, integrated within the EU Research Infrastructures CLARIN and DARIAH.

In addition to thanking the keynote speakers who accepted our invitation, we would like to thank all members of the Programme Committee and all additional reviewers. They ensured that the best papers were included in the Proceedings and provided invaluable comments to the authors.

We would like to use this paragraph to acknowledge the members of the Organising Committee, who worked very hard during the last few months and whose dedication and efforts made the organisation
of this event possible. All members of the Organising Committee (listed in alphabetical order below) carried out numerous organisational tasks and were eager to step in and support the organisation of the conference whenever needed: Dinara Akmurzina, Isuri Anuradha, Lucía Bellés-Calvera, Sol Berges, Maria Carmela Cariello, Rocío Caro Quintana, Ana Isabel Cespedosa Vázquez, Parthena Charalampidou, Anna Beatriz Dimas Furtado, Souhila Djabri, Anne Eschenbrücher, Marie Escribe, Darya Filippova, René Alberto García Taboada, Diana Geneva, Dinara Gimadi, Zara Kancheva, Alfiya Khabibullina, Lilis Kharatian, Shaifali Khulbe, Aida Kostikova, Sonia Kropiowska, Lydia Körber, Maria Kunilovskaya, Anastasia Laktionova, Ljubica Leone, Gabriela Llull, Jessica López Espejel, Ana Isabel Martínez-Hernández, Laura Mejías Climent, Alistair Plum, Kateryna Poltorak, Ivaylo Radev, Branislava Sandrih, Georgi Shopov, Nikola Spasovski, Natalia Sugrobova, Marina Tonkopeeva and Cecilia Valdenea. A big THANK YOU to all of you, this challenging in terms of organisation event could not have been taken place so smoothly without you!

Finally, many thanks go to the University of Wolverhampton, the Institute of Information and Communication Technologies at the Bulgarian Academy of Sciences, and Sirma AI for their generous support of RANLP.

Welcome to this year’s virtual event RANLP-2021 and we hope that you enjoy the conference!

The RANLP 2021 Organisers
The International Conference RANLP–2021 is organised by:

Research Group in Computational Linguistics,
University of Wolverhampton, UK

Department of Artificial Intelligence and Language Technologies,
Institute of Information and Communication Technologies,
Bulgarian Academy of Sciences, Bulgaria

RANLP–2021 is partially supported by:
Sirma AI

Programme Committee Chair:
Ruslan Mitkov, University of Wolverhampton, UK

Organising Committee Chair:
Galia Angelova, Bulgarian Academy of Sciences, Bulgaria

Workshop Coordinator:
Kiril Simov, Bulgarian Academy of Sciences, Bulgaria

Publishing Team:
Ivelina Nikolova-Koleva, Bulgarian Academy of Sciences and Sirma AI, Bulgaria
Georgi Shopov, Bulgarian Academy of Sciences, Bulgaria
Diana Geneva, Bulgarian Academy of Sciences, Bulgaria
Nikolai Nikolov, INCOMA Ltd., Shoumen, Bulgaria

Programme Committee Coordinators:
Ivelina Nikolova-Koleva, Bulgarian Academy of Sciences and Sirma AI, Bulgaria
Maria Kunilovskaya, University of Wolverhampton, UK
Rocío Caro, University of Wolverhampton, UK
Anne Eschenbrucher, University of Wolverhampton, UK
Maria Carmela Cariello, University of Wolverhampton, UK

The team behind RANLP-2021:
Galia Angelova, Bulgarian Academy of Sciences, Bulgaria
Ruslan Mitkov, University of Wolverhampton, UK
Preslav Nakov, Qatar Computing Research Institute, HBKU, Qatar
Nikolai Nikolov, INCOMA Ltd., Shoumen, Bulgaria
Ivelina Nikolova, Bulgarian Academy of Sciences, Bulgaria
Petya Osenova, Bulgarian Academy of Sciences, Bulgaria
Kiril Simov, Bulgarian Academy of Sciences, Bulgaria
Programme Committee:

Cengiz Acarturk (Middle East Technical University, Turkey)
Hassina Aliane (Research Centre on Scientific and Technical Information, Algeria)
Pascal Amsili (New Sorbonne University - Paris 3, France)
Le An Ha (University of Wolverhampton, United Kingdom)
Verginica Barbu Mititelu (Romanian Academy Research Institute for Artificial Intelligence, Romania)
Alberto Barrón-Cedeño (University of Bologna, Italy)
Riza Batista-Navarro (University of Manchester, United Kingdom)
Leonor Becerra (Aix-Marseille University, France)
Andrea Bellandi (National Research Council, Italy)
Victoria Bobicev (Technical University of Moldova, Moldova)
Svetla Boytcheva (Bulgarian Academy of Sciences, Bulgaria)
António Branco (University of Lisbon, Portugal)
Chris Brew (LivePerson, United States of America)
Erik Cambria (Nanyang Technological University, Singapore)
Burcu Can (University of Wolverhampton, United Kingdom)
María Carmela Cariello (University of Pisa, Italy)
Sheila Castilho (Dublin City University, Ireland)
Yue Chen (Indiana University, United States of America)
Parthena Charalampidou (Aristotle University of Thessaloniki, Greece)
Kenneth Church (Baidu, United States of America)
Kevin Cohen (University of Colorado, United States of America)
Orphee De Clercq (Ghent University, Belgium)
Radina Dobreva (University of Edinburgh, United Kingdom)
Anne Eschenbrücher (University of Wolverhampton, United Kingdom)
Antonio Ferrández Rodríguez (University of Alicante, Spain)
Alexander Gelbukh (National Polytechnic Institute, Mexico)
Tracy Holloway King (Adobe Inc., United States of America)
Veronique Hoste (Ghent University, Belgium)
Diana Inkpen (University of Ottawa, Canada)
Elena Irimia (The Romanian Academy, Romania)
Hitoshi Isahara (Otemon Gakuin University, Japan)
Udo Kruschwitz (University of Regensburg, Germany)
Sandra Kübler (Indiana University, United States of America)
Maria Kunilovskaya (University of Wolverhampton, United Kingdom)
Ladislav Lenc (University of West Bohemia, Czech Republic)
Natalia Loukachevitch (Lomonosov Moscow State University, Russia)
Stefano Marchesin (University of Padua, Italy)
Eugenio Martínez-Cámara (University of Granada, Spain)
Ricardo Muñoz Martín (University of Bologna, Italy)
Antonio Valerio Miceli Barone (University of Edinburgh, United Kingdom)
Johanna Monti (University of Naples “L’Orientale”, Italy)
Andrés Montoyo (University of Alicante, Spain)
Rafael Muñoz-Guillena (University of Alicante, Spain)
Preslav Nakov (Qatar Computing Research Institute, Qatar)
Roberto Navigli (Sapienza University of Rome, Italy)
Ivelina Nikolova-Koleva (Bulgarian Academy of Sciences and Sirma AI, Bulgaria)
Maciej Ogrodniczuk (Polish Academy of Sciences, Poland)
Constantin Orasan (University of Surrey, United Kingdom)
Petya Osenova (Sofia University and Bulgarian Academy of Sciences, Bulgaria)
Vasile Pais (The Romanian Academy, Romania)
Pavel Pecina (Charles University, Czech Republic)
Maciej Piasecki (Wroclaw University of Science and Technology, Poland)
Jakub Piskorski (Polish Academy of Sciences, Poland)
Alistair Plum (University of Wolverhampton, United Kingdom)
Gábor Prószéký (Hungarian Research Centre for Linguistics and Pázmány Péter Catholic University, Hungary)
Tharindu Ranasinghe (University of Wolverhampton, United Kingdom)
Omid Rohanian (University of Oxford, United Kingdom)
Paolo Rosso (Polytechnic University of Valencia, Spain)
Raheem Sarwar (University of Wolverhampton, United Kingdom)
Nasredine Semmar (Laboratory for Integration of Systems and Technology, France)
Vilelmini Sosoni (Ionian University, Greece)
Yoshimi Suzuki (University of Yamanashi, Japan)
Stan Szpakowicz (University of Ottawa, Canada)
Hristo Tanev (Joint Research Centre, European Commission, Italy)
Dan Tufis (Romanian Academy, Romania)
Luis Alfonso Ureña López (University of Jaén, Spain)
Marcos Zampieri (Rochester Institute of Technology, United States of America)
Michael Zock (National Centre for Scientific Research, France)

Reviewers:
Noor Abo Mokh (Indiana University, United States of America)
Ahmed AbuRa’ed (Pompeu Fabra University, Spain)
Itziar Aldabe (University of the Basque Country, Spain)
Ahmed Amine Aliane (Research Centre on Scientific and Technical Information, Algeria)
Ekaterina Artemova (Higher School of Economics University, Russian Federation)
Iana Atanassova (University of Bourgogne Franche-Comte, France)
Eduard Barbu (Institute of Computer Science, Estonia)
Antonina Bondarenko (University of Paris, France)
Aurélien Bossard (Paris 8 University, France)
Boryana Bratanova (St. Cyril and St. Methodius University of Veliko Tarnovo, Bulgaria)
Aljoscha Burchardt (German Research Centre for Artificial Intelligence, Germany)
Lindsay Bywood (University of Westminster, United Kingdom)
Oliver Cakebread-Andrews (University of Wolverhampton, United Kingdom)
Pablo Calleja (Universidad Politécnica de Madrid, Spain)
Hiram Calvo (Centre for Computing Research, Mexico)
Rocío Caro Quintana (University of Wolverhampton, United Kingdom)
Thiago Castro Ferreira (Federal University of Minas Gerais, Brazil)
Daniel Dakota (Indiana University, United States of America)
Kareem Darwish (Qatar Computing Research Institute, HBKU, Qatar)
Angelo Mario Del Grosso (National Research Council, Italy)
Mattia A. Di Gangi (AppTek GmbH, Germany)
Asma Djaidri (University of Science and Technology Houari Boumediene, Algeria)
Kevin Deturck (ERTIM & Viseo Technologies, France)
Sobha Lalitha Devi (Anna University, India)
Marie Escribe (University of Wolverhampton, United Kingdom)
Richard Evans (University of Wolverhampton, United Kingdom)
Mariano Felice (University of Cambridge, United Kingdom)
Vasiliki Foufi (University of Geneva, Switzerland)
Thomas François (Université catholique de Louvain, Belgium)
Emma Franklin (University of Wolverhampton, United Kingdom)
Larissa Freitas (Federal University of Pelotas, Brazil)
Björn Gambäck (Norwegian University of Science and Technology, Norway)
Aina Garí Soler (University of Paris-Saclay, France)
Federico Gaspari (Dublin City University, Ireland)
Darina Gold (University of Duisburg-Essen, Germany)
Momchil Hardalov (Sofia University "St. Climent Ohridski", Bulgaria)
Ali Hatami (University of Wolverhampton, United Kingdom)
Tomáš Hercig (University of West Bohemia, Czech Republic)
Diliara Iakubova (Kazan Federal University, Russian Federation)
Adrian Iftene ("Alexandru Ioan Cuza” University of Iași, Romania)
Camelia Ignat (Joint Research Centre, European Commission, Italy)
Milica Ikonić Nešić (University of Belgrade, Serbia)
Dmitry Ilvovsky (National Research University Higher School of Economics, Russia)
Radu Ion (Research Institute for Artificial Intelligence, Romanian Academy, Romania)
Tomoya Iwakura (Fujitsu, Japan)
Rubén Izquierdo Bevia (Nuance Communications, Spain)
Héctor Jiménez-Salazar (Metropolitan Autonomous University, Mexico)
Danka Jokić (University of Belgrade, Serbia)
Olga Kanishcheva (National Technical University, Ukraine)
Georgi Karadzhov (University of Cambridge, United Kingdom)
Aleksandar Kartelj (University of Belgrade, Serbia)
Dimitar Kazakov (University of York, United Kingdom)
Payal Khullar (International Institute of Information Technology, India)
Yasen Kiprov (Sofia University "St. Kliment Ohridski", Bulgaria)
Olga Kolesnikova (National Polytechnic Institute, Mexico)
Yannis Korkontzelos (Edge Hill University, United Kingdom)
Venelin Kovatchev (University of Birmingham, United Kingdom)
Cvetana Krstev (University of Belgrade, Serbia)
Todor Lazarov (New Bulgarian University, Bulgaria)
Tommaso Lo Sterzo (National Research Council, Italy)
Elpida Loupaki (Aristotle University of Thessaloniki, Greece)
Mireille Makary (Lebanese International University, Lebanon, and University of Wolverhampton, United Kingdom)
Suresh Manandhar (Wiseyak, United States of America)
Michał Marcińczuk (Wrocław University of Science and Technology, Poland)
Patricia Martín-Chozas (Universidad Politécnica de Madrid, Spain)
Halyna Maslak (University of Wolverhampton, United Kingdom)
Irina Matveeva (Reveal, Unites States of America)
Lesly Miculicich (Microsoft, United States of America)
Todor Mihaylov (Facebook, United States of America)
Cristina Vertan (University of Hamburg, Germany)
Veronika Vincze (University of Szeged, Hungary)
Victoria Yaneva (University of Wolverhampton, United Kingdom)
Sevinj Yolchuyeva (Budapest University of Technology and Economics, Hungary)
Kristina Yordanova (University of Rostock, Germany)
Wajdi Zaghouani (Hamad Bin Khalifa University, Qatar)
Mai Zaki (American University of Sharjah, United Arab Emirates)
Carlos Mario Zapata Jaramillo (Universidad Nacional de Colombia, Colombia)
Alisa Zhila (Amazon, United States of America)
He Zhou (Indiana University, United States of America)
Inès Zribi (University of Sfax, Tunisia)

Operational Team Leaders:
Rocío Caro Quintana, University of Wolverhampton, UK (Website content)
Ana Isabel Cespedosa Vázquez, University of Malaga, Spain (Correspondence)
Aida Kostikova, New Bulgarian University, Bulgaria (Session chairs)
Lydia Körber, University of Potsdam, Germany (Zoom hosting)
Maria Kunilovskaya, University of Wolverhampton, UK (Conference programme)
Gabriela Llull, Argentine Catholic University, Argentina (Final paper check-up)
Ana-Isabel Martínez-Hernández, University Jaume I, Castellón, Spain (Conference documentation)
Alistair Plum, University of Wolverhampton, UK (Website maintenance)
Organising Committee:

Dinara Akmurzina, New Bulgarian University, Bulgaria
Isuri Anuradha, University of Wolverhampton, UK
Lucía Bellés-Calvera, University Jaume I, Castellón, Spain
Sol Berges, University of Buenos Aires, Argentina
Maria Carmela Cariello, University of Wolverhampton, UK
Parthena Charalampidou, Aristotle University of Thessaloniki, Greece
Anna Beatrix Dimas Furtado, University of Malaga, Spain
Souhila Djabri, University of Alicante, Spain
Anne Eschenbrücher, University of Wolverhampton, UK
Marie Escribe, University of Wolverhampton, UK
Darya Filipova, University of Wolverhampton, UK
René A. García Taboada, University of Wolverhampton, UK
Diana Geneva, Bulgarian Academy of Sciences, Bulgaria
Dinara Gimadi, University of Wolverhampton, UK
Zara Kancheva, Bulgarian Academy of Sciences, Bulgaria
Alfiya Khabibullina, Ghent University, Belgium
Lilit Kharatian, University of Wolverhampton, UK
Shaifali Khulbe, University of Wolverhampton, UK
Sonia Kropiowska, University of Wolverhamton, UK
Anastasia Laktionova, University of Wolverhamton, UK
Ljubica Leone, Lancaster University, UK
Jessica López Espejel, Université Sorbonne Paris Nord, France
Ana Isabel Martínez Hernández, University Jaume I, Castellón, Spain
Laura Mejías Climent, University Jaume I, Castellón, Spain
Kateryna Poltorak, University of Wolverhampton, UK
Ivaylo Radev, Bulgarian Academy of Sciences, Bulgaria
Branislava Sandrih, Belgrade University, Serbia and University of Wolverhampton, UK
Georgi Shopov, Bulgarian Academy of Sciences, Bulgaria
Nikola Spasovski, University of Wolverhampton, UK
Natalia Sugrobova, Ghent University, Belgium
Marina Tonkopeeva, University of Wolverhampton, UK
Cecilia Valdenea, University of Wolverhampton, UK
# Table of Contents

**BPoMP: The Benchmark of Poetic Minimal Pairs – Limericks, Rhyme, and Narrative Coherence**  
Almas Abdibayev, Allen Riddell and Daniel Rockmore ......................................................... 1

**Ontology Population Reusing Resources for Dialogue Intent Detection: Generic and Multilingual Approach**  
Cristina Aceta, Izaskun Fernández and Aitor Soroa ................................................................. 10

**Efficient Multilingual Text Classification for Indian Languages**  
Salil Aggarwal, Sourav Kumar and Radhika Mamidi ............................................................. 19

**A New Quality Estimation Approach to Machine Translation**  
Benyamin Ahmadnia and Bonnie Dorr ..................................................................................... 26

**Domain Adaptation for Hindi-Telugu Machine Translation Using Domain Specific Back Translation**  
Hema Ala, Vandan Mujadia and Dipti Sharma ....................................................................... 31

**ArabGlossBERT: Fine-Tuning BERT on Context-Gloss Pairs for WSD**  
Moustafa Al-Hajj and Mustafa Jarrar ....................................................................................... 40

**English-Arabic Cross-language Plagiarism Detection**  
Naif Alotaibi and Mike Joy ........................................................................................................ 49

**Towards a Better Understanding of Noise in Natural Language Processing**  
Khetam Al Sharou, Zhenhao Li and Lucia Specia .................................................................... 58

**Comparing Supervised Machine Learning Techniques for Genre Analysis in Software Engineering Research Articles**  
Felipe Araújo de Britto, Thiago Castro Ferreira, Leonardo Pereira Nunes and Fernando Silva Parreiras .......................................................................................................................... 68

**Enriching the Transformer with Linguistic Factors for Low-Resource Machine Translation**  
Jordi Armengol-Estapé, Marta R. Costa-jussà and Carlos Escolano ......................................... 78

**A Multi-Pass Sieve Coreference Resolution for Indonesian**  
Valentina Kania Prameswara Artari, Rahmad Mahendra, Meganingrum Arista Jiwanggi, Adityo Anggraquito and Indra Budi ................................................................................................... 84

**Solving SCAN Tasks with Data Augmentation and Input Embeddings**  
Michal Auersperger and Pavel Pecina ....................................................................................... 91

**PyEuroVoc: A Tool for Multilingual Legal Document Classification with EuroVoc Descriptors**  
Andrei-Marius Avram, Vasile Pais and Dan Ioan Tufis ................................................................ 97

**TEASER: Towards Efficient Aspect-based SEntiment Analysis and Recognition**  
Vaibhav Bajaj, Kartikey Pant, Ishan Upadhyay, Srinath Nair and Radhika Mamidi ................. 107

**Interactive Learning Approach for Arabic Target-Based Sentiment Analysis**  
Husamelddin Balla, Marisa Llorens Salvador and Sarah Jane Delany ....................................... 116

**Litescale: A Lightweight Tool for Best-worst Scaling Annotation**  
Valerio Basile and Christian Cagnazzo ..................................................................................... 126
Character-based Thai Word Segmentation with Multiple Attentions
Thodsaporn Chay-intr, Hidetaka Kamigaito and Manabu Okumura ........................................ 269

Are Language-Agnostic Sentence Representations Actually Language-Agnostic?
Yu Chen and Tania Avgustinova ................................................................. 279

Investigating Dominant Word Order on Universal Dependencies with Graph Rewriting
Hee-Soo Choi, Bruno Guillaume, Karën Fort and Guy Perrier .......................... 286

RED: A Novel Dataset for Romanian Emotion Detection from Tweets
Alexandra Ciobotaru and Liviu P. Dinu ..................................................... 296

Assessing the Eligibility of Backtranslated Samples Based on Semantic Similarity for the Paraphrase Identification Task
Jean-Philippe Corbeil and Hadi Abdi Ghavidel ........................................... 306

Fine-tuning Neural Language Models for Multidimensional Opinion Mining of English-Maltese Social Data
Keith Cortis, Kanishk Verma and Brian Davis .......................................... 314

Towards an Etymological Map of Romanian
Alina Maria Cristea, Anca Dinu, Liviu P. Dinu, Simona Georgescu, Ana Sabina Uban and Laurentiu Zoicas ........................................................... 320

A Syntax-Aware Edit-based System for Text Simplification
Oscar M. Cumbicus-Pineda, Itziar Gonzalez-Dios and Aitor Soroa ................. 329

On Generating Fact-Infused Question Variations
Arthur Deschamps, Sujatha Das Gollapalli and See-Kiong Ng ............................ 340

Event Prominence Extraction Combining a Knowledge-Based Syntactic Parser and a BERT Classifier for Dutch
Thierry Desot, Orphee De Clercq and Veronique Hoste .................................. 351

Automatic Detection and Classification of Mental Illnesses from General Social Media Texts
Anca Dinu and Andreea-Codrina Moldovan ................................................. 363

A Pre-trained Transformer and CNN Model with Joint Language ID and Part-of-Speech Tagging for Code-Mixed Social-Media Text
Suman Dowlagar and Radhika Mamidi ...................................................... 372

Tracing Source Language Interference in Translation with Graph-Isomorphism Measures
Koel Dutta Chowdhury, Cristina España-Bonet and Josef van Genabith ............. 380

Decoupled Transformer for Scalable Inference in Open-domain Question Answering
Haytham Elfdaeel and Stanislav Peshterliev .................................................. 391

Towards Task-Agnostic Privacy- and Utility-Preserving Models
Yaroslav Emelyanov ................................................................................... 399

Knowledge Discovery in COVID-19 Research Literature
Ernesto L. Estevanell-Valladares, Suilan Estevez-Velarde, Alejandro Piad-Morffis, Yoan Gutierrez, Andres Montoyo, Rafael Muñoz and Yudivian Almeida Cruz ................................. 407

Online Learning over Time in Adaptive Neural Machine Translation
Thierry Etchegoyhen, David Ponce, Harritxu Gete and Victor Ruiz ..................... 416
Application of Deep Learning Methods to SNOMED CT Encoding of Clinical Texts: From Data Collection to Extreme Multi-Label Text-Based Classification
Anton Hristov, Aleksandar Tahchiev, Hristo Papazov, Nikola Tulechki, Todor Primov and Svetla Boytcheva .................................................................562

Syntax Matters! Syntax-Controlled in Text Style Transfer
Zhiqiang Hu, Roy Ka-Wei Lee and Charu C. Aggarwal ........................................571

Transfer Learning for Czech Historical Named Entity Recognition
Helena Hubkrová and Pavel Kral ................................................................. 581

Personality Trait Identification Using the Russian Feature Extraction Toolkit
James R. Hull, Valerie Novak, C. Anton Rytting, Paul Rodrigues, Victor M. Frank and Matthew Swahn ................................................................. 588

Semi-Supervised Learning Based on Auto-generated Lexicon Using XAI in Sentiment Analysis
Hohyun Hwang and Younghoon Lee ............................................................ 598

Multiple Teacher Distillation for Robust and Greener Models
Artur Ilichev, Nikita Sorokin, Irina Piontkovskaya and Valentin Malykh .............. 606

BERT Embeddings for Automatic Readability Assessment
Joseph Marvin Imperial ................................................................. 616

Semantic-Based Opinion Summarization
Marcio Inácio and Thiago Pardo .............................................................. 624

Using Collaborative Filtering to Model Argument Selection
Sagar Indurkhya ................................................................. 634

Domain-Specific Japanese ELECTRA Model Using a Small Corpus
Youki Itoh and Hiroyuki Shinnou ............................................................. 645

BERT-PersNER: A New Model for Persian Named Entity Recognition
Farane Jalali Farahani and Gholamreza Ghasem-Sani ..................................... 652

Cross-lingual Fine-tuning for Abstractive Arabic Text Summarization
Mram Kahla, Zijian Győző Yang and Attila Novák ....................................... 660

Behavior of Modern Pre-trained Language Models Using the Example of Probing Tasks
Ekaterina Kalyaeva, Oleg Durandin and Alexey Malafeev ................................ 669

Towards Quantifying Magnitude of Political Bias in News Articles Using a Novel Annotation Schema
Lalitha Kameswari and Radhika Mamidi .......................................................... 676

Application of Mix-Up Method in Document Classification Task Using BERT
Naoki Kikuta and Hiroyuki Shinnou ............................................................. 684

Translation Memory Retrieval Using Lucene
Kwang-hyok Kim, Myong-ho Cho, Chol-ho Ryang, Ju-song Im, Song-yong Cho and Yong-jun Han 689

Now, It’s Personal: The Need for Personalized Word Sense Disambiguation
Milton King and Paul Cook ................................................................. 697
Multilingual Image Corpus: Annotation Protocol  
Svetla Koeva ......................................................... 706

ELERRANT: Automatic Grammatical Error Type Classification for Greek  
Katerina Korre, Marita Chatzipanagiotou and John Pavlopoulos .................. 713

Neural Machine Translation for Sinhala-English Code-Mixed Text  
Archchana Kugathasan and Sagara Sumathipala ...................................... 723

Multilingual Multi-Domain NMT for Indian Languages  
Sourav Kumar, Salil Aggarwal and Dipti Sharma .................................. 732

Fiction in Russian Translation: A Translationese Study  
Maria Kunilovskaya, Ekaterina Lapshinova-Koltunski and Ruslan Mitkov ....... 739

Corpus Creation and Language Identification in Low-Resource Code-Mixed Telugu-English Text  
Siva Subrahmanyam Varma Kusampudi, Anudeep Chaluvadi and Radhika Mamidi .......... 749

Sentiment Analysis in Code-Mixed Telugu-English Text with Unsupervised Data Normalization  
Siva Subrahmanyam Varma Kusampudi, Preetham Sathineni and Radhika Mamidi .......... 758

From Constituency to UD-Style Dependency: Building the First Conversion Tool of Turkish  
Aslı Kuzgun, Öğuz Kerem Yıldız, Neslihan Cesur, Büşra Marşan, Arifete Betül Yenice, Ezgi Sanryar, Oğuzhan Kuyrukçu, Bilge Nas Arçan and Olcay Taner Yıldız .................. 766

Making Your Tweets More Fancy: Emoji Insertion to Texts  
Jingun Kwon, Naoki Kobayashi, Hidetaka Kamigaito, Hiroya Takamura and Manabu Okumura775

Addressing Slot-Value Changes in Task-oriented Dialogue Systems through Dialogue Domain Adaptation  
Tiziano Labruna and Bernardo Magnini .............................................. 785

Developing a Clinical Language Model for Swedish: Continued Pretraining of Generic BERT with In-Domain Data  
Anastasios Lamproudis, Aron Henriksson and Hercules Dalianis .................... 795

Text Retrieval for Language Learners: Graded Vocabulary vs. Open Learner Model  
John Lee and Chak Yan Yeung ......................................................... 803

Transforming Multi-Conditioned Generation from Meaning Representation  
Joosung Lee ................................................................................. 810

Frustration Level Annotation in Latvian Tweets with Non-Lexical Means of Expression  
Viktorija Leonova and Janis Zuters ................................................ 819

System Combination for Grammatical Error Correction Based on Integer Programming  
Ruixi Lin and Hwee Tou Ng .......................................................... 829

Multilingual Learning for Mild Cognitive Impairment Screening from a Clinical Speech Task  
Hali Lindsay, Philipp Müller, Insa Kröger, Johannes Tröger, Nicklas Linz, Alexandra Konig, Radia Zeghari, Frans RJ Verhey and Inez HGB Ramakers ........................................ 835

Naturalness Evaluation of Natural Language Generation in Task-oriented Dialogues Using BERT  
Ye Liu, Wolfgang Maier, Wolfgang Minker and Stefan Ultes .......................... 844
Towards the Application of Calibrated Transformers to the Unsupervised Estimation of Question Difficulty from Text
Ekaterina Loginova, Luca Benedetto, Dries Benoit and Paolo Cremonesi ........................................... 851

GeSERA: General-domain Summary Evaluation by Relevance Analysis
Jessica López Espejel, Gaël de Chalendar, Jorge Garcia Flores, Thierry Charnois and Ivan Vladimir Meza Ruiz ................................................................. 861

On the Interaction between Annotation Quality and Classifier Performance in Abusive Language Detection
Holly Lopez Long, Alexandra O’Neil and Sandra Kübler ................................................................. 872

NEREL: A Russian Dataset with Nested Named Entities, Relations and Events
Natalia Loukachevitch, Ekaterina Artemova, Tatiana Batura, Pavel Braslavskii, Ilia Denisov, Vladimir Ivanov, Suresh Manandhar, Alexander Pugachev and Elena Tutubalina .................................................. 880

Active Learning for Interactive Relation Extraction in a French Newspaper’s Articles
Cyrielle Mallart, Michel Le Nouy, Guillaume Gravier and Pascale Sébillot ........................................ 890

ROFF - A Romanian Twitter Dataset for Offensive Language
Mihai Manolescu and Çağrı Çöltekin .................................................................................................. 899

Monitoring Fact Preservation, Grammatical Consistency and Ethical Behavior of Abstractive Summarization Neural Models
Iva Marinova, Yolina Petrova, Milena Slavcheva, Petya Osenova, Ivaylo Radev and Kiril Simov ......................................................................... 905

Cultural Topic Modelling over Novel Wikipedia Corpora for South-Slavic Languages
Filip Markoski, Elena Markoska, Nikola Ljubešić, Eftim Zdravevski and Ljupco Kocarev ........... 914

Discovery of Multiword Expressions with Loanwords and Their Equivalents in the Persian Language
Katarzyna Marszalek-Kowalewska ........................................................................................................ 922

The Impact of Text Normalization on Multiword Expressions Discovery in Persian
Katarzyna Marszalek-Kowalewska ........................................................................................................ 933

Improving Neural Language Processing with Named Entities
Kyoumoto Matsushita, Takuya Makino and Tomoya Iwakura ................................................................ 944

TREMoLo-Tweets: A Multi-Label Corpus of French Tweets for Language Register Characterization
Jade Mekki, Gwénot Lecorvé, Delphine Battistelli and Nicolas Béchet ........................................................................ 954

Ranking Online Reviews Based on Their Helpfulness: An Unsupervised Approach
Alimuddin Melleng, Anna Jurek-Loughrey and Deepak P ....................................................................... 963

incom.py 2.0 - Calculating Linguistic Distances and Asymmetries in Auditory Perception of Closely Related Languages
Marius Mosbach, Irina Stenger, Tania Avgustinova, Bernd Möbius and Dietrich Klakow .......... 972

Not All Linearizations Are Equally Data-Hungry in Sequence Labeling Parsing
Alberto Muñoz-Ortiz, Michalina Strzyz and David Vilares ........................................................................ 982

Pre-training a BERT with Curriculum Learning by Increasing Block-Size of Input Text
Koichi Nagatsuoka, Clifford Broni-Bediako and Masayasu Atsumi ................................................... 993

COVID-19 in Bulgarian Social Media: Factuality, Harmfulness, Propaganda, and Framing
Preslav Nakov, Firoj Alam, Shaden Shaar, Giovanni Da San Martino and Yifan Zhang ................ 1001

xix
Siamese Networks for Inference in Malayalam Language Texts
Sara Renjit and Sumam Mary Idicula ................................................................. 1171

A Call for Clarity in Contemporary Authorship Attribution Evaluation
Allen Riddell, Haining Wang and Patrick Juola .................................................. 1178

Varieties of Plain Language
Allen Riddell and Yohei Igarashi ................................................................. 1184

Word Discriminations for Vocabulary Inventory Prediction
Frankie Robertson .................................................................................. 1192

FrenLyS: A Tool for the Automatic Simplification of French General Language Texts
Eva Rolin, Quentin Langlois, Patrick Watrin and Thomas François .................. 1200

Spelling Correction for Russian: A Comparative Study of Datasets and Methods
Alla Rozovskaya ............................................................................. 1210

Sentiment-Aware Measure (SAM) for Evaluating Sentiment Transfer by Machine Translation Systems
Hadeel Saadany, Constantin Orăsan, Emad Mohamed and Ashraf Tantavy .............. 1221

Multilingual Epidemic Event Extraction: From Simple Classification Methods to Open Information Extraction (OIE) and Ontology
Sihem Sahnoun and Gaël Lejeune ............................................................... 1231

Exploiting Domain-Specific Knowledge for Judgment Prediction Is No Panacea
Olivier Salaün, Philippe Langlais and Karim Benyekhlef .................................. 1238

Masking and Transformer-based Models for Hyperpartisanship Detection in News
Javier Sánchez-Junquera, Paolo Rosso, Manuel Montes-y-Gómez and Simone Paolo Ponzetto 1248

Serbian NER&Beyond: The Archaic and the Modern Intertwined
Branislava Šandrih Todorović, Cvetana Krstev, Ranka Stanković and Milica Ikonić Nešić .... 1256

A Semi-Supervised Approach to Detect Toxic Comments
Ghivvago Damas Saraiva, Rafael Anchiêta, Francisco Assis Ricarte Neto and Raimundo Moura 1265

Graph-based Argument Quality Assessment
Ekaterina Saveleva, Volha Petukhova, Marius Mosbach and Dietrich Klakow ........... 1272

A Hybrid Approach of Opinion Mining and Comparative Linguistic Analysis of Restaurant Reviews
Salim Sazzed ........................................................................... 1285

A Lexicon for Profane and Obscene Text Identification in Bengali
Salim Sazzed ........................................................................... 1293

A Case Study of Deep Learning-Based Multi-Modal Methods for Labeling the Presence of Questionable Content in Movie Trailers
Mahsa Shafaei, Christos Smailis, Ioannis Kakadiaris and Thamar Solorio .................. 1301

A Domain-Independent Holistic Approach to Deception Detection
Sadat Shahriar, Arjun Mukherjee and Omprakash Gnawali .................................. 1312

Towards Domain-Generalizable Paraphrase Identification by Avoiding the Shortcut Learning
Xin Shen and Wai Lam ................................................................. 1322
AutoChart: A Dataset for Chart-to-Text Generation Task
Jiawen Zhu, Jinye Ran, Roy Ka-Wei Lee, Zhi Li and Kenny Choo . . . . . . . . . . . . . . . . . . . . . . . . . 1640

A Comparative Study on Abstractive and Extractive Approaches in Summarization of European Legislation Documents
Valentin Zmiycharov, Milen Chechev, Gergana Lazarova, Todor Tsonkov and Ivan Koychev . 1649

Not All Comments Are Equal: Insights into Comment Moderation from a Topic-Aware Model
Elaine Zosa, Ravi Shekhar, Mladen Karan and Matthew Purver . . . . . . . . . . . . . . . . . . . . . . . . . . 1656
BPoMP: The Benchmark of Poetic Minimal Pairs – Limericks, Rhyme, and Narrative Coherence

Almas Abdibayev
Department of Computer Science
Dartmouth College
Hanover, NH 03755
almas.abdibayev.gr@dartmouth.edu

Allen Riddell
Department of English
Indiana University
Bloomington, IN 47408
riddella@indiana.edu

Daniel Rockmore
Department of Computer Science
Dartmouth College
Hanover, NH 03755
daniel.n.rockmore@dartmouth.edu

Abstract

We adapt BLiMP (Benchmark of Linguistic Minimal Pairs) language model evaluation framework to the context of poetry, introducing the first of a series of tasks titled Benchmark of Poetic Minimal Pairs (BPoMP). The tasks presented herein use one genre of English-language poetry, the limerick (five-lines, rhyme scheme AABBA). Following the BLiMP schema, the BPoMP tasks use 10,000 minimal pairs of limerick/corrupted limerick. The latter is created by (1) shuffling two rhyming end-of-the-line words, (2) shuffling two rhyming lines, (3) replacing end-of-the-line word by a non-rhyming synonym. Our general task is detection of the original limerick, which we believe tests a language model’s capacity to utilize “end rhymes”, a common feature of poetry. We evaluate Transformer-based models by checking if they assign a higher probability to the non-corrupted limerick in each minimal pair. We find that the models identify the original limerick at rates better than chance, but with a nontrivial gap relative to human accuracy (average of 98.3% across tasks). The publicly available curated set of limericks accompanying this paper is an additional contribution. In general, we see this as a first step to create a community of NLP activity around the rigorous computational study of poetry.

1 Introduction

Machines — i.e., artificial intelligence — continue to make great progress toward the challenges of machine-assisted and machine-written literature. Examples range from word auto-completion algorithms to deep learning-based methods that (sometimes) produce Turing Test-passing text (Elkins and Chun, 2020) and even credible sonnets (Ghazvininejad et al., 2016). Evaluating such work uses a range of techniques, including automatic metrics such as BLEU as well as – rightly or wrongly – Turing Test-inspired human evaluation.

This paper focuses on testing if large models pretrained on gigabytes of text, such as GPT-2 (Radford et al., 2019) and Transformer-XL (Dai et al., 2019)), are capable of “discovering” the rhyming and basic narrative information inherently present within a simple poetic form, the limerick, a five-line poem, usually humorous, with rhyme scheme AABBA and regular – albeit not fixed – metrical structure. In particular, we test Transformer models on various (suitably contextualized) forms of the line completion problem for the limerick as well as a fundamental narrative-oriented task. These tests should be viewed as a first step toward a broader goal of framing the rigorous interrogation of the poetic (or more broadly, literary) capabilities of language models. Given their formulaic structure (e.g., AABBA rhyming scheme) and the relatively small size of each unit, limericks present a good testing ground as compared to longer or more complex poems, such as Shakespearean sonnets (see e.g., (Ghazvininejad et al., 2016)), or most broadly, “literature”.

To this end we gathered a small data set of 99,000 limericks (filtered down further for quality purposes), which we use to test three poetic feature: understanding the concept of a rhyme framed as two kinds of line completion test, and one test of narrative structure, framed as a line ordering test, given the full limerick. The primary contri-
butions of this paper are the instantiations of these tests as “minimal pair” tasks and a curated and publicly available dataset of limericks. The former are designed in the spirit of the BLiMP (Benchmark of Linguistic Minimal Pairs) tests (Warstadt et al., 2020) that are used to interrogate more general linguistic capabilities of language models. Our limerick-based test is a first construction that will be part of a larger collection of tests of machine poetry capabilities, given the title BPoMP, or Benchmark of Poetic Minimal Pairs. As for the latter contribution we hope that the availability of this curated data set inspires other computational studies of this and other poetic forms.

Recognizing the basic ingredients of a limerick (or any poetic form) are also part of being able to produce a limerick and thus, we also see this as a first step toward another larger goal: machine composition of high-quality limericks, or any prescribed poetic form.

2 Related Work

This paper is a part of the growing body of literature devoted to the formulation and execution of tasks for probing language models, including the “BERTology” literature that focuses on BERT (see e.g., (Tenney et al., 2019; Michel et al., 2019; Clark et al., 2019; Hewitt and Manning, 2019). In formulation, it is directly modeled on the BLiMP framework (Warstadt et al., 2020), wherein the machine is given a pair of instances, one correct, and another “minimally” modified (i.e., “corrupted” in some very minor, but systematic way) and the language model then outputs probabilities (or some derivative thereof) signifying which is the most likely. Our line completion tasks produce line completion-based metrics, and thus are related to the large body of work already devoted to the sentence completion task (see e.g., (Mirowski and Vlachos, 2015)). Our simple exploration of narrative structure, interrogated through the minimal pair of limericks, one an original and the other with two rhyming lines swapped, is related to, but different from work on next sentence prediction (Cui et al., 2018), which presents two sentences of text in original and swapped orders (but without the fuller context that our minimal pair of limericks provides) and produces probabilities for both occurrences.

Our work also drew partial inspiration from poetry generation literature (Ghazvininejad et al., 2016; Li et al., 2018; Liu et al., 2019; Lau et al., 2018). However, in this work we do not aim to create machine poetry, but rather show the way in which poetry can be used as a prism to evaluate models which purport to be accurate models of a wide variety of human-produced texts, i.e. texts written in a variety of genres and registers.

3 Problem Statement

In the spirit of BLiMP (Warstadt et al., 2020), we present the language model with a five-line limerick and its corrupted counterpart and compare the probabilities the model assigns to each of these text blocks. Note that for each test we always include a full limerick and full limerick with corruption (which thus may or may not still be a limerick). The corrupted limerick differs from its source in several possible ways (in the order of increasing difficulty): (1) two rhyming words swap places, (2) two rhyming lines swap places, (3) one word is changed, always an end-of-the-line rhyming word which has been replaced by a synonym (see below for details) that removes rhyming information. Each of the three defines its own task. The difficulty ordering is based on the amount of corruption introduced into the limerick—from somewhat obvious to more subtle. Note however, that this doesn’t necessarily correspond to empirical human rating of difficulty. We mostly focus on causal models, which allows us to infer the probability of sequence (i.e. limerick), by multiplying estimated probabilities of each token using previously seen tokens as context (in practice we sum log-probabilities) (Bengio et al., 2003):

$$P(S) = \prod_{i=0}^{|[S]|} P(w_i|w_{<i})$$

where $S$ is the limerick sequence and $w_i$ is the token at the position $i$.

In some cases (i.e. BERT) we use a modified pseudo-likelihood formulation we adapted from (Lau et al., 2020), to compute “probabilities” using bidirectional context:

$$P(S) = \prod_{i=0}^{|[S]|} P(w_i|w_{<i}, w_{>i})$$

This produces a simple experiment (and 3 tasks): given two sequences, a limerick and its corrupted
version, the model must correctly “guess” the original by assigning a higher probability to it. Each limerick pair then serves as a single test point. However, the source limerick may appear more than once within the dataset, since we may have generated several corrupted versions.

4 Experimental Setup

4.1 Dataset

We did not source published books of limericks, as they lacked machine-readable editions and tended to use dated language. Instead, we worked from the website The Omnificent English Dictionary In Limerick Form (OEDILF). OEDILF\(^1\), established in 2004, publishes user-submitted limericks subject to approval by moderators. The topics are generated by users and moderators. Our scrape of OEDILF contains limericks by 1624 different authors. The distribution of limericks by author is not uniform, with the top 10 authors responsible for ca. 40 % of all limericks. We first filtered limericks based on simple criteria: limericks must have 5 lines and must use words (as opposed to symbols, such as emojis or formulae). This excludes a range of unorthodox limericks such as ASCII art limericks. The result was a corpus of 98,454 limericks (dated 1998–present). For our experiment we further require all “limericks” satisfy some simple machine-identifiable limerick criteria: (a) end-of-the-line words must be in our rhyming dictionary, (b) 2nd and 3rd lines must have fewer syllables than other lines. Our first criterion is there to ensure all our limericks can be identified as following the AABBA rhyme scheme. Our second follows from the conventional definition of a limerick (and is tied to metrical structure). Verification uses CMUdict\(^2\) and a custom character-level RNN model used to predict the number of syllables in the words, trained on CMUdict and syllable count information from Wiktionary\(^3\). The details of this model can be found in the Appendix. This second stage filtering leaves us with 29,853 limericks to build the set of minimal pairs. This clean subset contains 52,316 unique words, and a total of 879,653 tokens. Next, we discuss the corruption process of the limericks for all of our tasks. Note that in every case, we sample a limerick from the clean subset and pair it up with its corrupted version using procedures described below.

4.2 “Shuffled” Test Sets

Two of our tasks shuffle the contents of the original limerick, by either swapping two words, or swapping two lines. We generated these as follows.

Our first shuffled dataset, samples a limerick and randomly swaps end-of-the-line two rhyming words (as defined by AABBA scheme) within it. Doing so, we aim to preserve rhyming information but also to distort the semantics. Swapped words are checked to belong to the same part of speech during the replacement process. In certain instances, rhyming words can in fact repeat within the same limerick, but we found that these account for only 0.05% of all altered limericks.

Our second shuffled dataset randomly swaps two lines in a sampled original limerick. In this case, we are trying to remove longer-horizon semantic information, as well as to disturb the meter. As previously, they must belong to the same part of rhyme scheme, A or B. We make sure to remove any obvious datapoints, such as lines that start a quotation, or corrupted limericks that now end with a non-line ending symbol, such as comma, as a result of the swap.

For both of these test sets, we sample 10,000 test

\(^1\)http://www.oedilf.com/db/Lim.php

\(^2\)http://www.speech.cs.cmu.edu/cgi-bin/cmudict

\(^3\)https://en.wiktionary.org/wiki/Wiktionary:Main_Page
pairs of original and altered limerick.

4.3 Corrupted Rhyme Test Set

The corrupted rhyming word limerick dataset was auto-generated using the following procedure. For each limerick we replace the $i$th end-of-the-line rhyming word, $i \in \{1, 2, 3, 4, 5\}$ with a non-rhyming synonym. To generate high-quality synonyms we used Merriam-Webster’s Collegiate Thesaurus API.\footnote{https://dictionaryapi.com/products/api-collegiate-thesaurus. We publish a static version of the set of minimal pairs so that all results reported here can be verified easily.} We were able to retrieve synonyms for 34,699 unique end-of-the-line words. Synonyms were then filtered by part of speech (Merriam-Webster provides different synonyms for each POS the query word potentially belongs to and it has in its database\footnote{For example eye returns ‘noun’: [‘ring’, ‘navel’, ‘scrutiny’, ‘...’, ‘consciousness’, ‘vision’, ‘loop’], ‘verb’: [‘deliberate’, ‘sight’, ‘ponder’, ‘meditate’, ‘perceive’, ‘...’, ‘watch’, ‘wrestle (with)’]}). We utilize spaCy (Honnibal et al., 2020) as our POS tagger. Once the set of synonyms is found, we choose the synonym with the closest contextualized representation (within the limerick) to the representation of an original word provided by BERT\textsubscript{LARGE} (Devlin et al., 2018). Separately, we score each synonym using a different scheme: we replace each end-of-the-line word with a synonym using the same process described above, and then use GPT-2 to score total limerick probability of each potential corrupted limerick. We then choose the highest probability replacement and see if it agrees with the BERT replacement. In case of agreement, we add the corrupted limerick to the pool. We want to stress that at no point are we comparing the corrupted limerick to the original in terms of total sequence probability as this would interfere with our experimental goals. In other words, all comparisons are made between candidate synonym replacements. This GPT-2 filtering ensures that most of our replacements are of high quality. Finally, we exclude all limericks whose tokens are not in any of the models’ vocabularies, and then sample 10,000 limerick pairs for the final test set.

4.4 Models

We assess a family of Transformer-based (Vaswani et al., 2017) language models. Transformer is a neural network that consists of several layers of attention (Bahdanau et al., 2014) interspersed with fully connected layers. For all Transformer models we utilize Hugging Face’s transformers (Wolf et al., 2020) library.

GPT-2 (Radford et al., 2019) is a causal language model, meaning that it predicts a word based only on preceding words. Thus, for line 3, the model uses all words from lines 1 and 2, and potentially uses rhyming information contained within the last word of line 1. We use GPT-2–Medium that has equivalent number of parameters (345 million) as all other Transformer models we test. GPT-2 was pretrained on a custom, 40 GB size dataset called WebText, which includes various HTML pages from around the web filtered so as not to include Wikipedia text.

BERT (Devlin et al., 2018) uses bidirectional self-attention, which in addition to looking behind, also looks ahead in sequence. BERT is trained on a masked language task, where in a given sequence (a sentence, a paragraph) certain words are masked at random and are then predicted given the surrounding context. We use uncased BERT in the generation of corrupted synonyms, and cased model in our experiments. In both cases we use 340M model. BERT is trained on a 3,300 million word combined corpus of regular prose (BookCorpus, (Zhu et al., 2015)) and an English Wikipedia dump scraped by the authors.

Transformer-XL (Dai et al., 2019) is another causal language model that differs through the addition of a recurrence mechanism: hidden states from previous segments (a fixed-length context window that the model uses to predict the next word) are carried over to the next segment, which theoretically extends the context length that Transformer-XL can utilize. In addition to support this recurrence mechanism the authors propose a novel position embedding scheme. Transformer-XL was pretrained on Wikitext-103 (Merity et al., 2016), a dataset of 103M tokens scraped from English Wikipedia. We use the 340 million parameter version of Transformer-XL.

XLNet (Yang et al., 2019) differs from other models in the optimization objective: the model is trained using a novel permutation language model objective, where instead of a fixed-order (left to right) context, the model is exposed to a randomly permuted sequence, while predicting the last word.
(or last few words) of this sequence. Since the new context includes tokens both from the left and right of the original context of the target word, the model is bidirectional. At the same time it is also causal as the generation is still left to right. In addition, XLNet reuses segment level recurrence as introduced in (Dai et al., 2019) to increase potential context length. XLNet, is trained on BookCorpus, English Wikipedia, but adds heavily filtered versions of datasets Giga5, ClueWeb 2012-B, and Common Crawl\(^6\) (Parker et al., 2011; Callan et al., 2009), resulting in 33 billion subword-piece dataset. The model we test has 340 million parameters.

5 Results

Our general experimental framework is as follows: we ask if the models assign higher probability to the uncorrupted limerick of the pair. Again, in each test the “minimal pair” is a pair of five-line texts, one the original limerick, the other a corruption, which may or may not still be a limerick. We compare our models using simple count-based accuracy: total number of correctly guessed pairs (i.e., pairs where the higher probability is assigned to the uncorrupted limerick) versus total number of (minimal) pairs.

Table 2 summarizes our results. For the task of differentiating between the original limerick and its twin with shuffled rhyming words, GPT-2 wins by a small margin over other Transformer models. The differences are very small, with the exception of Transformer-XL, which performed poorly (a pattern that appears in other tasks). Overall, the models were easily able to distinguish semantically inconsistent corrupted limericks.

Next we take a look at the task of swapping lines that rhyme. Since the semantic issues here are not as apparent, we expected performance of all models to dip compared to our first task. This largely has proven to be true, as accuracy drops by an average of 11% across all models. Here, as well, GPT-2 emerges to be superior in terms of performance.

Finally, we assess results on the most difficult task, where we preserve the semantic information but diminish rhyming information, by replacing one word with a non-rhyming synonym. The accuracy drops significantly, compared to first two tasks, and the gap between human and model performance is considerable. Here, BERT is the clear “winner” is that it outperforms the favorite GPT-2 by almost 5 percentage points. We theorize that the better performance comes from BERT’s increased bidirectional context. Based on these three experiments, rhyming information seems to matter to a non-trivial extent for these models.

5.1 Human Baseline

All models perform worse than human judges. Human readers correctly identify the original limerick with probability 1 in shuffled rhyming words and corrupted rhyming word tasks. The subjects emphasized that they identified corrupted limericks after scanning end-of-the-line words to find the non-rhyming word. On the shuffled line task they get 95% accuracy. The human experiments were performed with 11 test subjects in corrupted rhyming word task and 2 test subjects in the remaining two tasks. All test subjects were native English speakers and none poetry experts. Each test subject was presented with 20 randomly sampled pairs of limericks presented in randomized order. Arguably, in the shuffled line test, for both test subjects the mislabeled limerick was still syntactically - and metrically and rhyme scheme correct. We suspect that a similar test performed with a larger sample of human annotators would produce identical results.

5.2 GPT-2 Error Analysis: Synonym Corruption Case Study

For a more detailed error analysis we focus on the GPT-2 results within the corrupted rhyme task as the model is the clearest in terms of optimization task (next-word prediction) and performs well in most tasks. To analyze the GPT-2 errors we considered the distribution of log-probability differences between original and corrupted limericks, identifying tail cases with largest difference.\(^7\) This is a proxy for a measure of model prediction “confidence.”

We hypothesize that difference in the n-grams counts\(^8\) \(n \in \{2, 3, 4\}\) used in the last tokens of the corrupted line is strongly correlated with the log probability (the one model assigns to the entire limerick) difference between the limerick pair.

First we consider examples where the model assigned relatively high probability to the correct (original) limerick. We expect to see that the original bi-gram\(^9\) occurs with much higher frequency

\(\text{https://commoncrawl.org/}\)

\(\text{https://books.google.com/ngrams}\)

\(\text{We start with bi-grams, and in some cases used 3- and}\)
Transformer-based models succeed at a rate better than chance, but much worse than the perfect and near-perfect human baseline. The limerick dataset is made publicly available.

Future near-term contributions to the BPoMP framework include meter, assonance, and alliteration tested in more complex short poetry. While the limerick may appear to be a “simple” textual form, it is in that way a perfect kind of “model organism” for a more general study of poetry and even literature. A five-line poem of this form can still articulate a range of narrative structure and sophisticated literary devices, but is of a size that admits relatively easy modification and interrogation – e.g., through the minimal pairs framework. We hope that this paper and the concomitant data set release inspire such future work.

6 Conclusion

In this paper we have introduced a first instance of a Benchmark of Poetic Minimal Pairs (BPoMP), a framework for the testing of the poetic “knowledge” of a language model. Using a curated set of limericks we interrogate the abilities of four Transformer-based language models on three poetry-based tasks (end-of-the-line rhyme replacement, rhyming word swapping, and rhymed line swapping) framed as distinguishing a limerick from a minimally corrupted version. In each case the

Table 2: Accuracy in all 3 tasks in which we stress test Transformer models, with last line showing the average (2 subjects) human performance. In shuffled rhyming word task, corrupted limerick has two rhyming end-of-the-line words switch places. Shuffled rhyming lines, as the name implies, switches the position of two lines, e.g. line 1 and 5 trade places and corrupted limerick is rearranged in the line form 5,2,3,4,1. Lastly, in corrupted rhyming word task we choose one end-of-the-line word at random and replace it with a non-rhyming synonym found in Merriam-Webster dictionary.

<table>
<thead>
<tr>
<th>Model</th>
<th>Shuffled rhyming words</th>
<th>Shuffled rhyming lines</th>
<th>Corrupted rhyming word</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT (cased)</td>
<td>0.9414</td>
<td>0.8321</td>
<td>0.7034</td>
</tr>
<tr>
<td>GPT-2</td>
<td>0.9423</td>
<td>0.8336</td>
<td>0.6557</td>
</tr>
<tr>
<td>Transformer-XL</td>
<td>0.8724</td>
<td>0.6623</td>
<td>0.6188</td>
</tr>
<tr>
<td>XLNet</td>
<td>0.9310</td>
<td>0.8227</td>
<td>0.6741</td>
</tr>
<tr>
<td>Human</td>
<td>1.0000</td>
<td>0.9500</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

than the corrupted bi-gram, which would translate to higher difference in bi-gram counts. The Pearson correlation for seven tail end cases was 0.71. This suggests that bi-gram frequency difference is a moderately good predictor of the log-probability difference. Next, we considered examples in which the model was very confident and wrong. In all such cases the GPT-2 output could be explained by bi-gram counts, with correlation of 0.89 for five extreme tail cases. In other words, the corrupted bi-grams displayed much higher frequency than their correct (original) counterparts. Note however, that this proxy falls apart as correlation reduces to $-0.39$ once we include more cases where the model isn’t as confident. Thus, for high confidence scenarios bi-gram statistics explain the differences in probabilities the model assigns to either original or a fake.

References


Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. 2009. Clueweb09 data set.


Appendix

A.1 Compute Resources

For most of the model testing and synonym generation work we used 2 NVidia Titan X graphics cards with 12GB of VRAM, running CUDA 11.2. In parallel, we also used Google Colab with T4 instances that sport 16GB of VRAM and also run CUDA 11.2.

A.2 OEDILF Description

The limericks are organized according to two categories: (a) included words contained and (b) topics. The latter are tags provided by both the managers of the website and the people who submitted the limericks. The dictionary part of the website refers to these words that could be found within a limerick, that is for any word in this dictionary there is a corresponding limerick that contains this word.

A.3 Identifying Number Of Syllables

Not all words in the vocabulary of our dataset are present in CMUdict, which only includes 133,779 examples. The full 98,454 limerick dataset (recall that there were two filtering steps to get us from this full set to the set used in the experiments) produces a vocabulary of 110,650 words, of which 55% are not present in CMUdict. To estimate the number of syllables in the vocabulary we trained a small (∼ 9800 parameters) character-level RNN\(^\text{12}\). The model consists of an embedding layer, a one-layer LSTM, with hidden size of 30, and a final linear layer that outputs back to characters. We train our charLSTM to be a classifier over 20 classes where each class determines the number of syllables in a word. The number 20 was set as an upper bound of number syllables based on the fact that the longest syllable word known in dictionaries has 19 syllables. A simple regression model did not work, nor did it optimize very well, displaying mode collapse-like behavior (i.e., consistent prediction of the average value, which is not useful for our discrete scenario).

\(^{12}\)We based the code from https://github.com/spro/char-rnn-pytorch

Table 3: Example of bigrams we captured for the analysis. Bigrams are highlighted in color. The limerick comes from a tail case when the model is correctly very confident.

Initially, we trained using pure CMUdict and around 15,000 words were used as a validation set, with another 118,000 used as as a training set. For our test set we decided to predict the number of syllables within lines of 21 limericks (which we labeled manually), as it emulates our scenario more closely than pure prediction of syllables in unseen words. We removed all of the words present in limericks from the training set. After analysis we determined that CMUdict did not contain many words with more than 11 syllables, so we supplemented it with another source: Wiktionary, an open source dictionary with user-submitted definitions of words. Wiktionary contains statistics with regards to number of syllables per word\(^\text{13}\) that can be scraped quite easily. We were able to obtain training examples with up to 19 syllables. In total that gave us additional 40789 training examples. We trained the model using ADAM optimizer, setting learning rate to 0.001 until validation loss went below 0.2 (around 10-12 epochs depending on the optimization curve).

A.4 GPT-2 Analysis And Bi-gram Outliers

In this section we present various supporting materials we used in our analysis of cases where the model assigned high probability to a limerick within a test pair. Table 3 presents an example limerick pair from our test set, in which GPT-2 correctly guessed the original with high confidence.

\(^{13}\)https://en.wiktionary.org/wiki/Category:English_words_by_number_of_syllables
Table 4: Outliers: target end-of-the-line word and its preceding neighbor taken from original and corrupted limericks which GPT-2 labelled correctly, but whose bi-gram frequency is higher for the corrupted limerick. This violates our hypothesis that bi-gram frequency would be higher for the original limerick. Upon closer examination we realize that 3- and 4-grams still fit our hypothesis. If frequency is listed as NA then it was not found in enough documents within Google Ngram corpora. Bigger n-grams (compared across rows) are bolded. The outliers were found when analyzing set of log-probabilities farther from the tail of empirical distribution.

<table>
<thead>
<tr>
<th>Ngram</th>
<th>Original</th>
<th>Corrupted</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-gram</td>
<td>your senses 6e-07</td>
<td>your feelings 1.73e-06</td>
</tr>
<tr>
<td>3-gram</td>
<td>to your senses 1.5e-07</td>
<td>to your feelings 6.2e-08</td>
</tr>
<tr>
<td>4-gram</td>
<td>come to your senses 9.5e-08</td>
<td>come to your feelings NA</td>
</tr>
<tr>
<td>2-gram</td>
<td>your court 1.3e-07</td>
<td>your yard 1.4e-07</td>
</tr>
<tr>
<td>4-gram</td>
<td>ball’s in your court 6e-9</td>
<td>ball’s in your yard NA</td>
</tr>
<tr>
<td>2-gram</td>
<td>a time 4.8e-05</td>
<td>a moment 7.3e-05</td>
</tr>
<tr>
<td>3-gram</td>
<td>at a time 1.8e-05</td>
<td>at a moment 1e-06</td>
</tr>
<tr>
<td>2-gram</td>
<td>the hip 1.8e-06</td>
<td>the cool 4.2e-06</td>
</tr>
<tr>
<td>3-gram</td>
<td>from the hip 1.3e-07</td>
<td>from the cool 7.4e-08</td>
</tr>
<tr>
<td>2-gram</td>
<td>for broke 6.9e-08</td>
<td>for poor 1.3e-06</td>
</tr>
<tr>
<td>3-gram</td>
<td>go for broke 4.1e-08</td>
<td>go for poor 3.7e-10</td>
</tr>
<tr>
<td>2-gram</td>
<td>to cheese 4.1e-08</td>
<td>to trash 7.9e-08</td>
</tr>
<tr>
<td>3-gram</td>
<td>due to cheese 2.4e-10</td>
<td>due to trash 1.2e-10</td>
</tr>
</tbody>
</table>

Figure 1: Histogram plotting log-probability difference between original and corrupted limerick. Since we are plotting correctly identified limerick pairs we are taking the difference between \( \log P(\text{original}) - \log P(\text{corrupted}) \) which is positive. Note the tail cases that start around 15.12 mark.

Figure 2: Histogram plotting log-probability difference between original and corrupted limerick. Since we are plotting incorrectly identified limerick pairs we are taking the difference between \( \log P(\text{corrupted}) - \log P(\text{original}) \) which is positive. Note the barely visible tail cases that start around 12 mark.
Ontology Population Reusing Resources for Dialogue Intent Detection: Generic and Multilingual Approach

Cristina Aceta, Izaskun Fernández
TEKNIKER
Basque Research and Technology Alliance
Eibar, Spain
cristina.aceta@tekniker.es
izaskun.fernandez@tekniker.es

Aitor Soroa
CCIA Group
University of the Basque Country
San Sebastián, Spain
a.soroa@ehu.eus

Abstract

This work presents a generic semi-automatic strategy to populate the domain ontology of an ontology-driven task-oriented dialogue system, with the aim of performing successful intent detection in the dialogue process, reusing already existing multilingual resources. This semi-automatic approach allows ontology engineers to exploit available resources so as to associate the potential situations in the use case to FrameNet frames and obtain the relevant lexical units associated to them in the target language, following lexical and semantic criteria, without linguistic expert knowledge. This strategy has been validated and evaluated in two use cases, from industrial scenarios, for interaction in Spanish with a guide robot and with a Computerized Maintenance Management System (CMMS). In both cases, this method has allowed the ontology engineer to instantiate the domain ontology with the intent-relevant information with quality data in a simple and low-resource-consuming manner.

1 Introduction

Nowadays, the arrival of new technologies and virtual assistants (e.g., Siri, Alexa) prove that naturally interacting with different devices and applications is a reality that is constantly improving through time. In many contexts, such as industrial scenarios, this fact has contributed to an increasing demand of technologies that allow workers to naturally communicate (i.e., without relying on specific constructions or keywords) with industrial systems such as robots, machines or information systems so as to increase productivity and security (González-Docasal et al., 2020), among other aspects.

A common approach to support natural communication between humans and machines are dialogue systems. When the goal is for the system to perform a specific action requested by the user, task-based dialogue systems are used (Jurafsky and Martin, 2020). As stated in Jurafsky and Martin (2020), a wide range of modern commercial task-oriented dialogue systems are to some extent inspired in the Genial Understannder System (GUS) architecture in Bobrow et al. (1977). This architecture consists of frame structures, that model the situations to be invoked by the user –the intents– and the required slots –arguments– to be provided for each frame. To determine the frame to be selected for a specific request, an intent detection step is needed.

The main research goal is to develop a generic semantic-based task-oriented dialogue system inspired by GUS, in which both the dialogue process and the domain modelling are conceptualized in a generic manner by making use of ontologies\(^1\). The focus in this paper is the population process of the domain ontology to efficiently perform intent detection, minimizing human intervention and linguistic knowledge when adapting the dialogue system to specific use cases.

Communication based on natural language can use a variety of words and expressions to convey the same meaning. Therefore, a dialogue system that interacts with users must be able to recognize different words that evoke the same situation or refer to the same entity to perform successful intent detection\(^2\). In an ontology-based approach, to populate the domain ontology with this information, it is very important that the maximum number of these possibilities is considered. Doing so manually would be a high time- and resource-consuming task, and the tendency is to apply semi-automatic or automatic techniques (Makki, 2017; Benabbas et al., 2018). According to this tendency, this work presents a generic strategy that aims to overcome

\(^1\)The former is usually called dialogue ontology, and the latter domain ontology.

\(^2\)For the previous example, besides inform, the verb tell may also elicit the give information intent.
the challenge of instantiating the domain ontology with intents and their trigger words by semi-automatically exploiting existing lexical resources, without reducing the quality of the results and decreasing time and costs in the adaptation to different scenarios and applications. The result, thus, is a semi-automatic process which leverages existing Semantic Frames (Fillmore et al., 1976) and obtains the maximum number of words that trigger them in different languages —i.e., their translation equivalents—, so as to instantiate the domain ontology of the dialogue system approach previously described. This process has been validated in two use cases from industrial scenarios.

The rest of the paper is organised as follows: Section 2 will provide relevant Related Work. Section 3 will describe the ontology population strategy. Section 4 will present the validation and evaluation of the strategy through 2 use cases, and Section 5 will provide Conclusions and Further Work.

2 Related Work

There is a wide range of works in the literature that deal with intent detection, as it is one of the core tasks in the interpretation component in task-oriented dialogue systems (Gupta et al., 2019). In this sense, two main approaches to perform intent detection can be distinguished: on the one hand, approaches that rely on machine learning (ML) methods and, on the other hand, approaches that make use of ontologies. For both methodologies, a fair amount of data is needed and, thus, cannot cope with scenarios with limited or nonexistent labeled data, which implies the necessity of performing manual work, which is resource-consuming (Chen et al., 2017).

For machine-learning-based methods, several approaches can be observed in the literature that make use of traditional ML algorithms such as Support Vector Machines (Cortes and Vapnik, 1995) or logistic regression (Bishop, 2006). However, modern approaches employ deep learning (DL) methods (Louvan and Magnini, 2020; Chen et al., 2019). As mentioned previously, ML-based approaches require a large volume of data to perform training tasks, especially DL models.

For ontology-based methods, the population of the domain ontology to perform intent detection is generally based on patterns, which can be obtained automatically or semi-automatically. The formalization in Cassier et al. (2019), who have designed an intent detection system for emails in French, is inspired in semantic frames and the population of the domain ontology is based on a set of mainly automatically-generated patterns that allow to determine the words that trigger the intents. However, to obtain these patterns, a large amount of data was collected and processed. On the other hand, Quamar et al. (2020) present a dialogue system in which intent detection in health-related texts is performed. As in the previous case, the authors also rely on patterns to determine Key and Dependent concepts, which are considered to determine the intent in a sentence. Also, the information related to patterns is obtained combining expert knowledge and automatic methods.

Considering the previous remarks, intent detection seems to require either a large volume of data or manual work. In ontology-based approaches, for the former, the task of instantiating the domain ontology in low-resourced scenarios is severely limited and, for the latter, the use of manual expert knowledge is time and resource consuming. In this sense, a well-known challenge to the Semantic Web community is how to reduce the amount of manual work in the ontology population task. As noted by Kontopoulos et al. (2017), most approaches for ontology population make use of textual input and rely on natural language processing techniques to obtain the necessary knowledge to populate the ontology (Corcoglioniti et al., 2016; Makki, 2017), whereas approaches that rely on data that is structured at some degree—which are of interest for this work— are less common (Leshcheva et al., 2017; Kontopoulos et al., 2017). Considering this, the authors in Kontopoulos et al. (2017) use structured knowledge in Linked Data for ontology population.

The approach presented in this paper aims to overcome these challenges providing a methodology that could be used in scenarios with a lack of data with a relative minimal effort, without depending of expert linguistic work, and by making use of already available structured knowledge.

2.1 Resources

The concept of FrameNet originates from Frame Semantics (Fillmore et al., 1976), which is the linguistic theory that asserts that specific words evoke specific frames. In a nutshell, FrameNet constitutes a multilingual predicate resource that aims
to model situations (frames) and the words that elicit them (lexical units) in a comprehensive way, and also provides with the necessary actors that take part on them (Frame Elements) as a sort of a slot-modelling approach. In FrameNet, frames are not language-specific, and therefore are common in all language versions. That said, the data that varies across languages are, for each frame, the corresponding lexical units in the target language.

FrameNet, however, is not the only resource that provides predicate-related information in the literature. Other examples are VerbNet (Schuler, 2006) or PropBank (Kingsbury and Palmer, 2003). Each of these resources present characteristics that the rest of alternatives do not offer which, in the end, has generated the need to integrate them into a single repository. The result of this integration is Predicate Matrix (PM) (De Lacalle et al., 2014), in which data from different predicate (e.g. English FrameNet), lexical (e.g., WordNet (Miller, 1995)) and semantic (e.g. Suggested Upper Merged Ontology - SUMO (Pease et al., 2002)) resources is mapped, providing a “multilingual interoperable predicate lexicon” (De Lacalle et al., 2016).

Regarding the data integrated in PM, one of the lexical resources included is WordNet. WordNet (Miller, 1995) is a lexical knowledge base that stores nouns, verbs, adjectives and adverbs –and their corresponding senses– and groups them semantically into what are called synsets, which are described in Miller (1995) as “sets of cognitive synonyms”. In this sense, words are interlinked in semantic terms, considering each word’s senses. WordNet has been adapted to many languages and even some wordnets have been integrated into single resources, such as the Multilingual Central Repository –MCR–(Atserias et al., 2004). The MCR, which at the time of writing this paper is on version 3.0, aims to provide a powerful and rich multilingual lexical knowledge base (Atserias et al., 2004). This lexical knowledge base integrates data from multiple resources, including WordNet –in its 3.0 version– in six different language versions (English, Spanish, Catalan, Basque, Galician and Portuguese). Moreover, following the EuroWordNet architecture (Atserias et al., 2004), the MCR interconnects these wordnets using interlingual indices (ILI), also based in WordNet 3.0, for equivalent synsets in different languages. Furthermore, the repository has been enriched with ontological knowledge coming from semantic resources such as Base Level Concepts (Rosch, 1977), WordNet domains (Bentivogli et al., 2004) and SUMO\(^3\) (Pease et al., 2002; Guinovart et al., 2021). MCR synsets are also mapped in PM.

In terms of semantic data, among others, PM includes SUMO. SUMO is intended as an upper level ontology to serve as a “foundation for more specific domain ontologies” (Niles and Pease, 2001) by including a wide range of general-purpose terms that comprehensively cover different fields, such as Linguistics, Computer Science or Artificial Intelligence. Also, these terms are formally defined through axioms, which makes SUMO a specially enriched resource. Thus, the aim of SUMO is to provide a comprehensive, precisely defined term ontology. The mapping of SUMO tags with the rest of the elements in PM allows to establish a categorization between the different terms in it.

It is worth noting that PM is centred in verbal information. Thus, for resources that account for multiple parts of speech (such as nouns or adjectives in WordNet or frames that have nominal lexical units in FrameNet), only verbal data is integrated, so as to be able to properly map the data from all the resources involved.

3 Generic Strategy for Ontology Population for Intent Detection

The aim of the generic strategy is to support ontologists in the process of task-oriented dialogue system domain ontology population, minimizing the manual work, reusing linguistic resources, but without expert knowledge of the linguistic field.

The core linguistic resource used for collecting the necessary information for the ontology population process is PM, since it integrates and maps, among others, the information contained in FrameNet, SUMO and the MCR—the resources that include most of the relevant data for this work.

The strategy will guide the ontologist through the selection of the relevant information within PM, and will gather, semi-automatically, the relevant intents (through frames), the associated trigger words (through lexical units) and their corresponding synonyms for a specific use case in which the dialogue will take place. Considering this, the main steps of the strategy are the following:

0. **Use Case Characterization.** The identification of the different situations (associated

---

3Equivalence (“=”), subsumption (“+”) and instantiation (“@”) mapping relations (Alvez et al., 2019).
that surround the use case is performed. For example, for a guide robot, the situations involved may be to be taken to a given place or just to present information.

1. **Linguistic-resource-driven data selection.**
   For the given situations identified for the use case, a data selection from linguistic resources is performed, following three phases: **frame selection, lexical unit selection** and **semantic extension and filtering**.

2. **Automatic data gathering for intents.** Considering the selected data, the relevant knowledge to the intents is automatically gathered and structured in terms of the intents, their corresponding frames and the words that evoke them that are relevant to the use case. **Up to this point, the gathered information is agnostic to the structure of the target domain ontology.**

3. **Query generation and ontology population.** Once the relevant intent-related information is automatically obtained, the query for the final population of the target domain ontology will be generated.

The following sections will detail the aforementioned steps.

### 3.1 Linguistic-resource-driven data selection

This first step is aimed to obtain the necessary information that will serve as input for the automatic data gathering step. This data will be selected according to the situations identified for the use case.

#### 3.1.1 Frame Selection

The first stage in the process is to select the most suitable frames for the previously defined events in the use case. For this step, the ontology engineer will choose the relevant frames for the use case from the list of available FrameNet frames in PM.

For example, in a guide robot scenario – described in more detail in Section 4.1-- – one frame to be chosen would be **Motion**.

#### 3.1.2 Lexical Unit Selection

For each selected frame, a set of lexical units in English are automatically extracted from PM and presented to the ontology engineer, who will select the relevant lexical units according to the use case. This step is necessary because it is not evident that all lexical units are suitable to all the cases the chosen frame may apply to.

To illustrate this with an example, the frame chosen in the previous section, **Motion**, applied to the guide robot use case, in which a robot may be asked to guide the user to some final destination, has the lexical units *fn:blow.v, fn:fly.v, fn:go.v, fn:move.v,* etc., associated in PM. Within the use case, some lexical units such as *fn:fly.v* do not apply, and must therefore not be selected. In this case, the relevant lexical units to be selected are *go.v* and *move.v*.

#### 3.1.3 Semantic Extension and Filtering

The objective of this step is to extend semantically the previously selected lexical units by exploiting MCR synsets and SUMO tags.

For that, firstly, an automatic semantic extension is performed through the links from lexical units to MCR synsets in PM, collecting all the synsets related to given lexical units. However, not all the synsets in MCR for a lexical unit may apply to the use case, because polysemous words may acquire different meanings in a same frame, so a selection is required. For example, *go* may imply movement from one place to another or to pass away.

In PM, synsets are mapped to SUMO tags, which aim to group different synsets into single concepts. Since SUMO tags are more human-readable than synsets and the amount of tags is considerably lower without losing significance, the selection of synsets is proposed to be performed through SUMO tags. For that, the associated SUMO tags for the whole synset list are automatically extracted from PM and presented to the ontology engineer for the final selection.

To continue with the previous example, for *fn:go.v,* the **mcr:SubjectiveAssessmentAttribute, mcr:OccupationalRole; duration, mcr:Motion** and **mcr:Death** SUMO tags are presented to the ontology engineer, from which they will select the most relevant ones for the use case (all except **mcr:Death**).

### 3.2 Automatic Data Gathering for Intents

The objective of this step is to obtain the final words in the target language that evoke the desired intents. For this, the translation equivalents corresponding to the set of synsets obtained through the data in the previous steps will be retrieved automatically.
Taking as input the set of selected frames, lexical units and SUMO tags, the corresponding synsets will be automatically retrieved from PM. After that, the words in the target language that belong to these synsets will be obtained from the MCR, also automatically. As a result of this step, a set of synonyms in the target language(s) for each lexical unit will be obtained.

In the example used previously, for the frame *Motion* and the relevant synsets for the lexical unit *go.v*, the resulting synonyms in Spanish are the following: *acudir, desplazarse, ir, mover, moverse, viajar* and *partir*.\(^5\)

At this point, all the necessary data to support intent detection has been obtained.

### 3.3 Query Generation and Ontology Population

In this step, the SPARQL query necessary to populate the ontology with the gathered data is created according to the modelling of the target ontology. To summarise, Figure 1 represents the complete process, including the example use case shown in the previous sections, which is described in more detail in Section 4.1.

### 4 Validation and Evaluation

The process has been implemented as an API REST and has been tested in two use cases: a guide robot scenario and a Computerized Maintenance Management System (CMMS), both for the Spanish language. The domain ontology used is the Domain Module for Task-Oriented Dialogue management Ontology (TODO) –TODODom–\(^6\), as it is an ontology that provides a top model for domain modelling for task-based dialogue systems. The implementation of the methodology to instantiate the ontology described in this work applied to these use cases will be detailed in the following lines.

#### 4.1 Use case #1: Interacting with a Guide Robot in Spanish

In this use case, the user needs to communicate to a guide robot in Spanish. This robot is able to guide the user to their destination of choice and to offer information from specific elements (a room, a machine, etc.). The situations that may be identified from this use case, thus, are: to move from one point to another, and to ask for information or details about something. After checking the FrameNet frames in PM, the ones deemed more relevant in this use case are the following: *Motion, Taking* and *Arriving*\(^7\).

The next step is to select the appropriate lexical units for each frame in PM. In this case, the total number of lexical units for all the frames selected in the previous step are 34. Although many of the lexical units do cover the use case, such as *move* and *arrive*, there are others, such as *fly* or *blow*, that are too specific and do not apply to the use case and must not be selected.

After this selection, the number of lexical units in English is reduced to 12: *go, move (Motion); take (Taking); approach, arrive, come, enter, get, make, reach, return, visit (Arriving)*.

So as to semantically filter the synsets associated to each lexical unit, SUMO tags are used. In this case, the number of available SUMO tags is 34. In this use case, the application of semantic filtering is specially relevant, since from the initial 34 available tags, only two are applicable to the use case and have been selected: *BodyMotion* and *Motion*. The rest of tags were related to situations that were not considered in the use case, such as *Death or Cooking* and, thus, were not part of the final selection. This considerable reduction of tags reinforces the importance of this filtering step.

Finally, after having selected the frames, lexical units in English and SUMO tags, this information is used to automatically extract the applicable synsets and their translation equivalents in Spanish which, in this case, make a total of 32. The vast majority of the synonyms are relevant to the use case (e.g., *moverse, acudir, desplazarse, regresar*\(^8\)), although a very small portion of them are not entirely appropriate (e.g., *estar_ activo – to be active*).

#### 4.2 Use case #2: Interacting with a CMMS in Spanish

In this use case, the target system is a Computerized Maintenance Management System (CMMS), in which the user should be able to ask about problem solving protocols, request blueprints and similar tasks. As the previous use case, the language used is Spanish. Considering the previous remarks, the

---

5*Attend, move, go, move, travel and leave*

6*https://w3id.org/todo/tododom*

7Although the original FrameNet includes the frame *Information*, which would suit this use case, it is not included in PM due to the fact that it only has nominal lexical units. In the context of this work, this frame will remain out of the evaluation experiments carried out in Section 4.3.

8*move, go, move and return, respectively.*
Figure 1: Diagram representing the ontology population process presented in this work in a guide robot scenario.

Table 1: Number of translation equivalents for each PM selected data configuration and use case.

<table>
<thead>
<tr>
<th>Guide</th>
<th>CMMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold</td>
<td>25</td>
</tr>
<tr>
<td>F</td>
<td>667</td>
</tr>
<tr>
<td>F+LU</td>
<td>170</td>
</tr>
<tr>
<td>F+SUMO</td>
<td>139</td>
</tr>
<tr>
<td>F+LU+SUMO</td>
<td>32</td>
</tr>
</tbody>
</table>

Identifiable situations in this use case are showing information and reporting and solving problems. The applicable frames are, thus, Resolve problem, Evidence, Reporting and Communication.

Regarding the lexical units, a total of 30 results have been obtained from the previously selected frames. After discarding some lexical units that were not applicable to the current use case, such as contradict or deal, the selected lexical units are the following: solve, resolve (Resolve проблем); indicate, reveal, show (Evidence and Communication); inform, report, tell (Reporting).

In this next step, the number of remaining SUMO tags is 9, which are reduced to Communication, VisualAttribute and IntentionalPsychologicalProcess. Given this selection, the total number of automatically obtained word synonyms in Spanish for this use case is 39, being most of them appropriate to the use case (e.g. solucionar, informar, presentar). However, other results are not applicable to the use case, such as denunciar (report, in the legal sense of the word).

4.3 Evaluation

The results obtained in the 2 use cases above have been evaluated to determine the suitability of the strategy described in this work. For this, a gold standard has been created using expert knowledge, by considering the synonyms in the target language (in this case, Spanish) for all the lexical units corresponding to the selected frames for each use case.

For this evaluation, different combinations of the manual selection steps in Section 3 have been applied, and their corresponding synonyms have been obtained through the resulting synsets. The combinations evaluated are the following:

- **Frame selection (F).** All lexical units for the selected frames and their associated synonyms.
- **Frame + lexical unit selection (F+LU).** Selection of lexical units for the selected frames.
- **Frame selection + SUMO filtering (F+SUMO).** All lexical units for the selected frames, plus SUMO filtering.
- **Frame+lexical unit selection+SUMO filtering (F+LU+SUMO).** Selection of lexical units for the selected frames, plus SUMO filtering. It is the strategy defined in this work.

Table 1 shows the number of synonyms obtained in each combination\(^\text{10}\). For each of the configurations, precision-recall-F1 evaluations have been performed, the results of which can be seen in Table 2. These evaluations show that the less filtering, the more recall, as the more synonyms obtained, the more possibility they will include the gold ones. In terms of F1, on average, unique filtering configurations (F+LU and F+SUMO) obtain practically the same results. It is especially relevant that SUMO filtering is able to obtain on average better precision.

\(^{10}\)Data can be found at [https://git.io/JB2HY](https://git.io/JB2HY).
<table>
<thead>
<tr>
<th></th>
<th>Guide</th>
<th></th>
<th>CMMS</th>
<th></th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>F1</td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>F</td>
<td>0.04</td>
<td>1</td>
<td>0.07</td>
<td>0.12</td>
<td>1</td>
</tr>
<tr>
<td>F+LU</td>
<td>0.13</td>
<td>0.88</td>
<td>0.23</td>
<td>0.23</td>
<td>0.94</td>
</tr>
<tr>
<td>F+SUMO</td>
<td>0.14</td>
<td>0.76</td>
<td>0.23</td>
<td>0.21</td>
<td>0.88</td>
</tr>
<tr>
<td>F+LU+SUMO</td>
<td>0.50</td>
<td>0.64</td>
<td>0.56</td>
<td>0.28</td>
<td>0.64</td>
</tr>
</tbody>
</table>

Table 2: Precision (P), recall (R) and F1 metrics for all configurations for each use case, and average results.

results than LU selection --+0.07 points--, which emphasises the importance of a semantic filtering step in this type of task. Nevertheless, it is important to point out that both configurations help improve the results. In this sense, both processes combined are complementary, since their combination allows to significantly improve the base results obtained with F.

All in all, these results show that manual filtering is necessary in this type of task, as it has been proved that improves up to 0.3 points the F1 measure on average. Also, the figures obtained allow to determine that this is a complex task that deserves further investigation.

### 4.4 Other remarks

The information gathered in the use cases considered in this work allows to instantiate the domain ontology with intent-detection-relevant data reducing effort and potential intent identification errors. As it can be seen from the data reported in Sections 4.1 and 4.2, the incremental filtering through the defined steps allows to drastically reduce options, and semantic filtering allows to fine-tune the lexical units to be associated to an intent.

As a side note, and regarding both use cases, manual selection is not always a straight-forward process. Regarding frames, the online version of FrameNet\(^1\) --with the corresponding description of each frame in human-friendly terms-- may be used to be able to differentiate ambiguous frames.

### 5 Conclusions

This work focuses on the intent detection step in an ontology-based task-oriented dialogue system, which requires a modelling of the intents into the domain ontology. Since the intents and the words that trigger them vary depending on the use case, the task of instantiating the domain ontology with such information every time a new use case is needed is time and cost consuming.

The paper has presented a generic, semi-automatic strategy that reuses existing multilingual lexical resources to assist ontology engineers in modelling the specific intents and the words that elicit them for a given use case. This method consists of a preliminary characterization of the use case, a selection of relevant FrameNet frames, lexical units in English and SUMO tags and an automatic data gathering for intents given the previously selected data. The result is a structured set of intents, their corresponding frames and the words that evoke them. Finally, and depending on the modelling of the target domain ontology, a SPARQL query is generated, to be used to instantiate it.

To prove the generality of the approach, this strategy has been validated in 2 different industrial use cases: interaction in Spanish with a guide robot and with a CMMS. Moreover, an evaluation in terms of precision, recall and F1 has been performed. This evaluation validates the steps described in this work and their positive contribution to instantiate valid intent-related information with a lower resource consumption than doing so manually.

Further work includes research oriented to the improvement of the quality of the data extracted. As stated previously, PM only includes frames that have verbs as lexical units, leaving out other frames, such as Information, that have nominal or adjectival lexical units. In this line, it may be interesting to explore other sources of information that allow to obtain information about these missing frames and in different languages, in a similar manner that PM does with frames with verbal lexical units.

### Acknowledgements

This work was partially supported by the Basque Government’s Elkartek R&I program, projects EKIN (KK-2020/00055) and DeepText (KK-2020/00088).

---

\(^1\)http://sato.fm.senshu-u.ac.jp/frameSQL/fn2_15/notes/
References


F. Corcoglioniti, M. Rospocher, and Alessio Palermo Aprosio. 2016. Frame-Based Ontology Population with PIKES. IEEE Transactions on Knowledge and Data Engineering, 28:3261–3275.


Abstract

India is one of the richest language hubs on the earth and is very diverse and multilingual. But apart from a few Indian languages, most of them are still considered to be resource poor. Since most of the NLP techniques either require linguistic knowledge that can only be developed by experts and native speakers of that language or they require a lot of labelled data which is again expensive to generate, the task of text classification becomes challenging for most of the Indian languages. The main objective of this paper is to see how one can benefit from the lexical similarity found in Indian languages in a multilingual scenario. Can a classification model trained on one Indian language be reused for other Indian languages? So, we performed zero-shot text classification via exploiting lexical similarity and we observed that our model performs best in those cases where the vocabulary overlap between the language datasets is maximum. Our experiments also confirm that a single multilingual model trained via exploiting language relatedness outperforms the baselines by significant margins.

1 Introduction

Text classification is the task of assigning pre-defined categories to free-text documents with the use of natural language processing (NLP). Here, the classifier is fed a text and it returns a category based on the content. For the purpose of this paper, the task is to classify whether a piece of news article is regarding sports or not. This process of assigning tags or categories to text according to its content helps businesses automatically structure and analyze text quickly and cost-effectively to automate processes and enhance data-driven decisions. With the growth of the Internet around the world, users write comments in different languages. But the majority of current classification systems still address only a single language, mainly English (Dashtipour et al., 2016). This increases the risks of missing essential information in texts written in other languages. Also, training these systems require substantial amounts of annotated datasets, which is again an arduous task for many languages. Same is the case with Indian languages. Despite having a very large number of native speakers, most of the Indian languages are still considered to be resource poor. There are not enough datasets available in most of the domains. Therefore, the task of text categorization becomes challenging for Indian languages.

Therefore, in order to classify data in different languages, multilingual text classification techniques are the need of the hour. Training a multilingual model would refrain us from training a separate model for different languages and it also helps the system in better learning by means of parameter sharing. This approach mainly works by combining all the data in hand and studies in machine translation (MT) have shown that multilingual learning is not much efficient in case of unrelated languages (Kudugunta et al., 2019; Kunchukuttan and Bhattacharyya, 2020). But this is not the case with Indian languages. Underlying the vast diversity in Indian languages are many commonalities. Because of contact over thousands of years, most of the Indian languages have undergone convergence to a large extent (Sridhar, 1981). These languages share many common words which have the same root word and meaning. However, they use different scripts derived from the ancient Brahmi script (Kunchukuttan and Bhattacharyya, 2020), but correspondences can be established between equivalent characters across different scripts. So, the main question arises whether we can benefit
from the relatedness found in between Indian languages? By relatedness, we refer to languages that exhibit lexical and structural similarities on account of sharing a common ancestry.

Thus, in this work, we put our efforts in exploring the zero-shot as well as multilingual text classification via exploiting lexical similarity of Indian languages. For zero-shot classification, we are proposing the efficient way of reusing a classification model trained on one language on some other Indian language. In addition to this, we also tackled the problem of deciding which language model to use for zero-shot classification for a particular test language. Our results confirm that maximum accuracy is achieved in those cases where the vocabulary overlap between the two language datasets is maximum. For efficient multilingual text classification, we are exploiting the lexical similarity via two techniques namely unified transliteration and subword segmentation. Our experiments also confirm that in case of low resource related languages, multilingual models achieve better accuracy than the baseline models.

This paper is further divided into 5 sections. Section 2 discusses related work in this area. Section 3 elaborated the methodology behind the different techniques and experiments. Section 4 elaborates the experimental details including the dataset preparation, dataset pre-processing and the experimental setup for training our models. All the results and analysis have been discussed in Section 5. Section 6 talks about conclusion and possible future work.

2 Related Work

One of the main problems in multilingual classification is the significant lack of resources (Balahur and Turchi, 2012). Thus, analysis in multiple languages is often addressed by transferring knowledge from resource-rich to resource-poor languages (Denecke, 2008), because there are no resources available in other languages. Much of the work in subjectivity analysis has been applied to English data, though work on other languages is growing: e.g., Japanese data are used in (Kobayashi et al., 2004; Suzuki et al., 2006; Kanayama and Nasukawa, 2006), German data are used by Kim and Hovy (2006b). Lexical approaches for sentiment analysis necessitate language specific lexical and linguistic resources. Generating these resources is very time consuming and often requires a lot of manual work. Methods have been developed for the mapping of subjectivity lexicons to other languages. To this aim, Kim and Hovy (2006a) use a machine translation system and subsequently use a subjectivity analysis system that was developed for English to create subjectivity analysis resources in other languages.

Another approach in obtaining subjectivity lexicons for other languages than English was explored by Banea et al. (2008b). In this work, authors attempt to leverage on the resources available for English and, by employing machine translation, generate resources for subjectivity analysis in other languages. This paper introduces a method for creating a subjectivity lexicon for languages with scarce resources. Further on, another approach to building lexicons for languages with scarce resources is presented by Banea et al. (2008a). This method is able to build a subjectivity lexicon by using a small seed set of subjective words, an online dictionary, and a small raw corpus, coupled with a bootstrapping process that ranks new candidate words based on a similarity measure. Machine translation for multilingual text classification has also seen attention from researchers. The approach is to use a machine translation system to translate texts in other languages into English: the text is translated from the original language into English, and then English-language resources such as SentiWordNet are employed (Denecke, 2008). Kanayama et al. (2004) translated only sentiment units with a pattern based approach. Balahur and Turchi (2014) used uni-grams, bi-grams and tf-idf features for building support vector machines on translated text. Boyd-Graber and Resnik (2010) built Latent Dirichlet allocation models to investigate how multilingual concepts are clustered into topics. Translation systems, however, have various problems, such as sparseness and noise in the data (Balahur and Turchi, 2012). Sometimes, the translation system does not translate essential parts of a text, which can cause serious problems, possibly reducing well-formed sentences to fragments. Therefore, we put our efforts for training a multilingual classifier without
India is known as the land of many tongues (Kunchukuttan and Bhattacharyya, 2020). There is no single language called “Indian”. India speaks hundreds of languages and dialects (Sengupta and Saha, 2015). Some are extinct, while some are still in use with considerable speakers. Despite having a lot of different scripts, most of the Indian languages still share a lot of lexical features and common words which have the same root and meaning which can be utilized to help improve the quality of zero-shot as well as text classification systems trained on them. So, in this paper we propose our technique of performing zero-shot classification via exploiting the language relatedness. Also, we investigate how multilingual classification models perform in case of Indian languages. To do this efficiently, we exploited the lexical similarity via two techniques namely unified transliteration and subword segmentation.

3.1 Exploiting Lexical similarity

Unlike the original multilingual text classification techniques which mostly aim at transfer learning via parameter initialisation i.e. learning from one high resource language and then transferring knowledge to some low resource language, we are exploiting lexical similarity between related Indian languages via parameter sharing. For this, we combined the two different approaches namely unified transliteration and subword segmentation to ensure that there is sufficient overlap between the vocabularies of the related Indian languages datasets.

3.1.1 Unified Transliteration

Since the languages involved in the models have different scripts, the data processing should help to map them into a common single script. So here, we transliterate all the Indian languages into a common Devanagari script (which in our case is script for Hindi) to share the same surface form (Kunchukuttan and Bhattacharyya, 2020). This unified transliteration is a string homomorphism, replacing characters in all the languages mentioned above with Hindi characters (script conversion to Devanagari) or consonant clusters independent of context.

3.1.2 Subword Segmentation

Despite sharing a lot of cognates, Indian languages do not share many words at their non-root level. Therefore, the more efficient approach is to exploit Indian languages at their sub-word level which will ensure more vocabulary overlap. Therefore, we are converting every word to sub-word level using the very well known technique Byte Pair Encoding (BPE) (Sennrich et al., 2015). This technique is applied after the unified transliteration in order to ensure that languages share same surface form (script). BPE units are variable length units which provide appropriate context for translation systems involving related languages. Since their vocabularies are much smaller than the morpheme and word level models, data sparsity is also not a problem. In a multilingual scenario, learning BPE merge rules will not only find the common sub-words between multiple languages but it also ensures consistency of segmentation among each considered language pair.

3.2 Zero-Shot Text Classification

There are many languages in India and one can not expect annotated data available in all of the domains for all of the languages. So in zero-shot text classification, the model can classify any text between given labels without any prior training data. For performing it efficiently for Indian languages, we are using the vocabulary overlap technique as discussed in Section 3.2.1. From our experiments, we noticed that the zero-shot classification performs best in those cases where the vocabulary overlap is maximum between the different language datasets. That model will perform best on that language, which is most similar to the

Figure 1: Multilingual Text Classification Pipeline
Table 1: Vocabulary Overlap Matrix

<table>
<thead>
<tr>
<th>Language</th>
<th>pa</th>
<th>gu</th>
<th>mr</th>
<th>or</th>
<th>bn</th>
<th>ta</th>
<th>te</th>
<th>ml</th>
<th>kn</th>
</tr>
</thead>
<tbody>
<tr>
<td>pa</td>
<td>-</td>
<td>67.87</td>
<td>71.34</td>
<td>58.35</td>
<td>55.77</td>
<td>38.55</td>
<td>61.27</td>
<td>54.16</td>
<td>61.87</td>
</tr>
<tr>
<td>gu</td>
<td>41.54</td>
<td>-</td>
<td>88.85</td>
<td>70.84</td>
<td>61.21</td>
<td>46.45</td>
<td>75.02</td>
<td>65.60</td>
<td>75.90</td>
</tr>
<tr>
<td>mr</td>
<td>18.32</td>
<td>37.28</td>
<td>-</td>
<td>35.08</td>
<td>28.36</td>
<td>55.34</td>
<td>70.35</td>
<td>59.27</td>
<td>76.14</td>
</tr>
<tr>
<td>or</td>
<td>32.27</td>
<td>64.02</td>
<td>75.53</td>
<td>-</td>
<td>60.30</td>
<td>50.21</td>
<td>67.46</td>
<td>67.35</td>
<td>69.57</td>
</tr>
<tr>
<td>bn</td>
<td>42.77</td>
<td>76.70</td>
<td>84.69</td>
<td>83.61</td>
<td>-</td>
<td>46.89</td>
<td>70.98</td>
<td>66.43</td>
<td>75.45</td>
</tr>
<tr>
<td>ta</td>
<td>11.75</td>
<td>23.13</td>
<td>65.68</td>
<td>27.67</td>
<td>18.63</td>
<td>-</td>
<td>75.23</td>
<td>79.43</td>
<td>77.24</td>
</tr>
<tr>
<td>te</td>
<td>18.46</td>
<td>36.93</td>
<td>82.52</td>
<td>36.74</td>
<td>27.88</td>
<td>74.35</td>
<td>-</td>
<td>82.00</td>
<td>92.77</td>
</tr>
<tr>
<td>ml</td>
<td>14.11</td>
<td>27.92</td>
<td>60.12</td>
<td>31.73</td>
<td>22.57</td>
<td>67.89</td>
<td>70.91</td>
<td>-</td>
<td>77.81</td>
</tr>
<tr>
<td>kn</td>
<td>11.80</td>
<td>23.65</td>
<td>56.54</td>
<td>23.99</td>
<td>18.76</td>
<td>48.33</td>
<td>-</td>
<td>58.73</td>
<td>56.97</td>
</tr>
</tbody>
</table>

training language dataset using vocabulary overlap technique. The method of calculating vocabulary overlap matrix is explained in the further subsection.

3.2.1 Vocabulary Overlap

We calculated similarity scores for each language based on vocabulary overlap by considering other language training data as monolingual data. Vocabulary Overlap provides a crude measure of surface form similarity between two languages. It is efficient to calculate, and is often quite effective, especially for low-resource languages. Here, we use the number of tokens that two languages share to measure the similarity between them. First, we transliterated each language to a common script, in our case it is Devanagari script and then all the tokens were converted into their respective subwords. After that score was calculated using the percentage of the common tokens in the two languages as:

\[
\text{vocab}_{l1-l2} = \frac{|\text{Token}_{l1} \cap \text{Token}_{l2}|}{|\text{Token}_{l1}|} \times 100
\]

The details of the vocabulary overlap matrix is shown in Table 1.

3.3 Multilingual Text Classification

We trained a Multilingual model, which is a single model that can handle multiple languages simultaneously. This would circumvent having to train a monolingual model for every single Indian language. One example in our case would be to classify whether a piece of news article is regarding sports or not. Using a regular Machine learning or Deep learning model, we would be able to classify only Punjabi language sports articles but not articles written in other Indian languages say Marathi. But if we use a multilingual model, we would be able to classify news articles in Punjabi, Marathi and multiple other Indian languages. Our results also prove that multilingual models achieve better performance than monolingual models, especially for low-resource languages. This could be the ability to learn not just from the training data of the language in question, but also from other language datasets.

But we noticed that this learning is not much efficient in case of the languages that don’t show any kind of relatedness. But on the other hand, Indian languages exhibit a lot of lexical and structural similarities on account of sharing a common ancestry. It is therefore important to utilize the lexical similarity of these languages to build systems by combining all the related languages. For the scope of this paper, we have trained 2 kinds of multilingual models. One with the training data of all the languages combined in their respective scripts and the other in which all the languages are first transliterated into one common script, in our case in Devanagari and then they are combined. To the best of our knowledge, this is the first time when someone has exploited the lexical similarity found in between Indian languages by using the techniques of unified transliteration and subword segmentation for the task of text classification. The pipeline is shown in Figure 1.

4 Experiments

4.1 Dataset

We have used IndicNLP News Article Classification Dataset (Kunchukuttan, 2020) for performing our experiments. This classification dataset comprises news articles and their categories for 9 different Indian languages. The dataset is balanced.
Table 2: Results of Zero-Shot Text Classification

<table>
<thead>
<tr>
<th>Language</th>
<th>pa</th>
<th>gu</th>
<th>mr</th>
<th>or</th>
<th>bn</th>
<th>ta</th>
<th>te</th>
<th>ml</th>
<th>kn</th>
</tr>
</thead>
<tbody>
<tr>
<td>pa</td>
<td>-</td>
<td>64.88</td>
<td>72.97</td>
<td>56.69</td>
<td>54.96</td>
<td>55.28</td>
<td>56.75</td>
<td>55.13</td>
<td>57.79</td>
</tr>
<tr>
<td>gu</td>
<td>64.78</td>
<td>-</td>
<td>74.10</td>
<td>61.44</td>
<td>61.51</td>
<td>54.31</td>
<td>66.56</td>
<td>60.19</td>
<td>68.00</td>
</tr>
<tr>
<td>mr</td>
<td>58.60</td>
<td>60.66</td>
<td>-</td>
<td>59.20</td>
<td>59.46</td>
<td>55.60</td>
<td>64.88</td>
<td>53.78</td>
<td>69.29</td>
</tr>
<tr>
<td>or</td>
<td>58.53</td>
<td>69.43</td>
<td>74.43</td>
<td>-</td>
<td>65.06</td>
<td>46.34</td>
<td>52.57</td>
<td>55.28</td>
<td>70.53</td>
</tr>
<tr>
<td>bn</td>
<td>49.40</td>
<td>56.47</td>
<td>63.79</td>
<td>56.22</td>
<td>-</td>
<td>47.13</td>
<td>54.91</td>
<td>51.68</td>
<td>53.24</td>
</tr>
<tr>
<td>ta</td>
<td>46.03</td>
<td>54.44</td>
<td>59.28</td>
<td>52.72</td>
<td>55.78</td>
<td>-</td>
<td>57.82</td>
<td>64.96</td>
<td>55.49</td>
</tr>
<tr>
<td>te</td>
<td>58.31</td>
<td>58.53</td>
<td>62.47</td>
<td>57.57</td>
<td>60.21</td>
<td>61.03</td>
<td>-</td>
<td>59.04</td>
<td>74.65</td>
</tr>
<tr>
<td>ml</td>
<td>52.60</td>
<td>56.03</td>
<td>58.50</td>
<td>55.10</td>
<td>62.44</td>
<td>58.51</td>
<td>63.21</td>
<td>-</td>
<td>75.78</td>
</tr>
<tr>
<td>kn</td>
<td>56.13</td>
<td>69.49</td>
<td>69.49</td>
<td>63.19</td>
<td>65.62</td>
<td>60.06</td>
<td>72.63</td>
<td>58.52</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3: Experimental results of all compared models.

<table>
<thead>
<tr>
<th>Model</th>
<th>pa</th>
<th>gu</th>
<th>mr</th>
<th>or</th>
<th>bn</th>
<th>ta</th>
<th>te</th>
<th>ml</th>
<th>kn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>93.69</td>
<td>95.01</td>
<td>93.97</td>
<td>95.45</td>
<td>94.26</td>
<td>95.82</td>
<td>96.05</td>
<td>92.48</td>
<td>94.15</td>
</tr>
<tr>
<td>Multilingual without</td>
<td>95.82</td>
<td>96.66</td>
<td>95.84</td>
<td>96.77</td>
<td>95.23</td>
<td>97.17</td>
<td>96.94</td>
<td>94.36</td>
<td>94.72</td>
</tr>
<tr>
<td>Multilingual with</td>
<td>97.66</td>
<td>98.77</td>
<td>97.49</td>
<td>97.10</td>
<td>96.46</td>
<td>97.77</td>
<td>98.33</td>
<td>95.83</td>
<td>96.99</td>
</tr>
</tbody>
</table>

4.2 Dataset Preprocessing

We noticed that the dataset contains a lot of punctuation, so we used manually created regex for cleaning the entire corpora. For all of our experiments, we have taken an equal number of training sentences from all the languages in order to maintain uniformity while training multilingual models. We have used the Indic NLP library (Kunchukuttan, 2020) for tokenization and normalization as our pre-processing steps. Also, to make sure that all the languages share the same surface form and have sufficient vocab overlap, we again used Indic NLP library (Kunchukuttan, 2020) for unified transliteration and subword segmentation.

4.3 Training Details

The dataset was split into a ratio of 4:1 for the purpose of training and testing. All of our experiments were performed using 5-fold cross-validation. We used an embedding layer followed by a Bidirectional LSTM layer followed by a dense layer. The embedding size was set to 128 and 128 hidden units were used in the LSTM layer. All the models were trained for 20 epochs with a batch size of 64. All of the experiments were performed using Keras library (Chollet et al., 2015) with Tensorflow (Abadi et al., 2015) as its backend.

5 Results and Analysis

Table 2 shows the results of zero-shot text classification on various languages. We can see that the maximum accuracy is achieved in those cases where the vocabulary overlap between the training and the monolingual dataset is maximum. For eg, Marathi model will perform best on the Punjabi test set due to maximum overlap. This can be confirmed from our vocabulary overlap matrix shown in Table 1. The reason is that most of the Indian languages share a lot of common sentiment bearing words and we exploit this using converting everything into same script and every word to its component subwords.
Table 3 shows the results of baselines as well as our multilingual models. We can see from the table that both the multilingual models outperform the baseline models. The reason is the ability to learn not only from its own dataset, but also from the other related languages present in the dataset. The best accuracy is achieved in the case where we apply the technique of unified transliteration and subword segmentation. The reason as explained above is the increase in vocabulary overlap which further increases the accuracy as compared to the first case. So, we can observe that in both the cases, we have benefitted from the lexical similarity found in Indian languages.

6 Conclusions and Future Work

In this work, we explored effective methods to exploit lexical similarity between related Indian languages in order to improve the quality of classification systems on low resource Indian languages. Our results confirm that a model trained on one Indian language can be reused for another Indian language and will provide good results if there is a significant vocabulary overlap between the two datasets. We also proved that for low resource Indian languages, multilingual models outperform the baseline models. The reason is the ability to learn not just from the training data of the language in question, but also from the other language datasets. Unified Transliteration and Subword Segmentation further increase the accuracy by increasing the vocab overlap among the datasets. Also, to get more increase in accuracy, one can try with large size datasets. For future work, we will try to apply this technique to other NLP related tasks for Indian languages.

References


Francois Chollet et al. 2015. Keras.


A New Quality Estimation Approach to Machine Translation

Benyamin Ahmadnia\textsuperscript{1} and Bonnie Jean Dorr\textsuperscript{2}

\textsuperscript{1}Brigham and Women’s Hospital and Harvard Medical School, Boston, MA, United States
\textsuperscript{2}Florida Institute for Human-Machine Cognition, Ocala, FL, United States
bahmadniyebosari@bwh.harvard.edu, bdorr@ihmc.org

Abstract

Procedures that perform the task of Machine Translation (MT) quality estimation are typically trained with text from the same distribution on which their performance is later evaluated. In most cases, additional human-scored source-target sentence pairs are provided for training. In this paper, we investigate an MT quality estimation setting where the algorithm has no access whatsoever to the test distribution of text or to an annotated training set with human judgment scores. Our method is based on a new notion of sentence cohesiveness. Our experiments on standard competition datasets for various language pairs show that the performance of our system is comparable to the baseline system.

1 Introduction

Procedures that perform the task of Machine Translation (MT) quality estimation are typically trained with texts from the same distribution on which their performance is later evaluated. In most cases, additional human scored source-target sentence pairs are provided for training. One flavor of the MT quality estimation task is to provide an MT metric like BLEU (Papineni et al., 2002) but without using reference translations.

The typical setting is a supervised learning one (Kreutzer et al., 2015; Martins et al., 2016, 2017), to mention a few. The algorithm is trained with a large corpus of source-target pairs, along with human judgment scores. The test set is typically sampled from the same distribution of the train. The algorithm is evaluated by comparing, for example, the scores that it assigns to the test sentence pairs with human judgment scores. When only text is provided for training, but no human judgment scores, then it is referred to as an unsupervised learning setting. Examples of unsupervised algorithms include Moreau and Vogel (2012); Banchs and Li (2011); Yankovskaya et al. (2019); Xu et al. (2020). This still does not preclude training on the same distribution of text as the test. Thus, the algorithm can use valuable features such as n-gram statistics or word embedding, tailored to that specific distribution.

Currently, there is no formal name or distinction from the scenario where the algorithm has no access whatsoever to the text distribution on which its performance is being tested and in particular no human judgment scores are provided. In this paper, we make this distinction and define the latter one as our learning setting. This setting makes sense in cases where access to the text’s distribution is impossible or too expensive. Furthermore, our learning setting can serve as a benchmark to test the robustness of an MT quality estimation algorithm to various degrees of noise in the training step.

The contribution of this paper is making formal the distinction between unsupervised and our proposed MT quality estimation, and providing a new quality estimation algorithm inspired by physical systems of particles. We view a sentence as a small system of interacting components (the words, represented using word embedding). For each sentence, we compute a cohesiveness factor, $N$, which reflects the extent to which the meaning of the entire system is a function of the meanings of its constituents. In fact, we take the difference in cohesiveness between source and target sentences as the measure for translation quality. The difference in cohesiveness is some aspect of adequacy, which is commonly understood as the amount of information (meaning) preserved between the reference and the candidate translation.

The proposed method is compatible with the learning setting as it can use word embedding that were trained on a generic text from the relevant source-target language pair. Furthermore, this
method utilizes only monolingual data that makes the method attractive in cases where cross-lingual parallel corpora is scarce or non-existent.

This paper is organized as follows; Section 2 details the proposed methodology. Section 3 outlines the experimental design and describes the results. Section 4 concludes the paper.

2 Methodology

An instance of the MT quality estimation problem is a pair of sentences $S$ (source) and $T$ (target). The output is a score that the algorithm assigns for the quality of the translation from $S$ to $T$ ($S \rightarrow T$). The words of $S$ are represented by word embedding, obtained e.g. via word2vec. These words are stacked into a sentence matrix $M_S$, whose rows are the $d$-dimensional vectors of the words.

To measure the extent to which the meaning of a sentence is a function of its words, we need to represent a sentence. The standard possibility is to train a system to represent all the words in a corpus as vectors. The purpose of the Skip-Gram Architecture is to train a Phrase-Based Statistical MT (PBSMT) decoder.

The $i$th row of $M_S$ (the $i$th word) can be written as:

$$W_i = \sum_{n=1}^{m} U_{in} \sigma_n v_n = U_{i1} \sigma_1 v_1 + \text{err}$$

(1)

That is, the vector of each word in the sentence is composed of the semantic contribution from $v_1$ (the main direction of the sentence) plus an error term, $\text{err}$. Thus, we can derive cohesiveness as the ratio:

$$n(M) = \frac{\sigma_1}{\sum_{i=1}^{m} \sigma_i}$$

(2)

For illustration consider the following three 5-word sentences: Dear dear dear dear dear (same word), Breakfast, dinner, lunch, milk, egg (same theme), and Monster, factory, gym, lake, chair (random themes). Using the vectors of Mikolov et al. (2013) we computed the sentence matrix and cohesiveness factor of each sentence. The first sentence scored $N = 1$, the second $N_2 = 0.46$ (nearly 50%) and the third merely $N_3 = 0.22$, which is roughly the expected approximation rate had $v_1$ been a random vector. In fact, the meaning of the first and second sentences is the “sum of their parts” (large cohesiveness) while the meaning of the third sentence is not (small cohesiveness).

Mathematically, the procedure receives as input the source sentence $S$, the target $T$, word embedding $w_S$ in the source language and $w_T$ in the target one, and an error $L : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, the better the translation. For the evaluation, we use:

$$L(x, y) = \max \{x, y\} / \min \{x, y\}$$

(3)

Algorithm 1 Proposed quality estimation

Procedure: quality estimation ($S$, $T$, $w_S$, $w_T$, $L$)

1: Embed $S$ and $T$ using the word-embedding $w_S$ and $w_T$ respectively
2: $M_S \leftarrow$ sentence matrix of $S$
3: $M_T \leftarrow$ sentence matrix of $T$
4: return $L(N(M_S), N(M_T))$

For the task of document-level quality estimation, the algorithm is applied iteratively on the sentences, and the final score is computed as the average of the $L$-values.

3 Evaluation

In our experiments, we employ “Google’s word2vec” (Mikolov et al., 2013) for English, and “Wikipedia-trained vectors” for non-English. All vectors are 300-dimensional, and also are trained using the skip-gram architecture with negative sampling.

The first batch of tests compares the performance of our proposed quality estimation algorithm (at sentence-level) against various supervised algorithms. As seen in Table 1, the first four columns show the results along with information about the baseline and best and supervised systems in that competition. As evident from Table 1, the performance of the supervised baseline which having access to human-judgment annotated data from the same distribution of they estimation algorithm train) is on average by merely 25% better.

The WMT’12 quality estimation task dataset (Callison-Burch et al., 2012) consists of 442 English-Spanish news texts produced by Moses4

4https://github.com/Kyuhong/wordvectors

4The purpose of the Skip-Gram Architecture is to train a system to represent all the words in a corpus as vectors.

4A Phrase-Based Statistical MT (PBSMT) decoder.
were manually annotated for quality in terms of post-editing effort (1-5 scores).

The WMT’17 quality estimation task dataset contains English sentences that are translated to German by various MT-systems, and ranked by correlation with HTER labels that are computed using TERCOM. We took 479 sentences translated by SYSTRAN4847. This system had the largest number of human scored sentence pairs.

The Japanese-English and Japanese-Chinese sentence pairs are taken from Fujita and Sumita (2017). The dataset contains 1676 sentences in Japanese that are obtained from role-play dialogues of healthcare providers. The sentences are translated into English and Chinese using their in-house MT system and quality is graded on a 1–5 scale, reflecting post-edit effort.

The baseline in the third share task of WMT’19 (Fonseca et al., 2019), LASER, is an unsupervised algorithm (Artetxe and Schwenk, 2019), but also supervised-learning algorithms competed in that task. The last two columns of Table 1 show how our approach compares on that data. The results that we reported are on a sample from the train set that was published by the organizers. We sampled about 200 sentences, which is roughly the size of the test set in that same competition (the test set, along with human-judgment scores, was not released). As evident, the proposed quality estimation outperforms LASER.

In the second batch of tests, we compared our method against BLEU at the document-level using two datasets sampled from two very different distributions. The first set consists of 100 online news pieces in English from websites like CNN, NBCNews and NYTimes. The second contains 100 English poems by more than 30 different poets written in the 19th century. The average number of words per poem is 130 and 196 for a news piece. For each text we performed forward translation into each of German and French and then back to English using three MT systems: Bing, Google and SDL. All texts and translations are provided as supplementary data.

We evaluated each pair using the proposed quality estimation and BLEU, the original text serving as a reference for BLEU. For each of the 200 pairs, we recorded the winning MT system according to BLEU and the proposed quality estimation. Tables 2 and 3 contain the ranking of the three MT systems. It shows that BLEU and the proposed quality estimation are aligned in their ranking: Bing > Google > SDL for German, French, and both distributions. The Spearman rank correlation between BLEU and the proposed quality estimation is around 0.4 for poetry (across languages), 0.27 for news via German and 0.16 for news via French.

We checked how sensitive are our results to the choice of text that was used to train the word embedding, and to the parameters of the training (specifically, with or without negative sampling). To this end, we repeat the tests with various combinations of pre-trained vectors, using either Google news text or Wikipedia. For English text, we replaced Google’s word2vec with WikiNeg\(^5\) and for non-English we replaced WikiNeg with WikiNorm.

Table 4 summarizes the correlation with human judgment scores when our proposed quality estimation was parameterized with the various com-

\(^5\)It means without negative sampling
Table 3: Comparing three MT systems over 100 English poems and 100 news pieces. Each cell is the number of times that MT system got the best score of the three according to BLEU or proposed.

<table>
<thead>
<tr>
<th>Systems</th>
<th>Gr-poetry-BLEU</th>
<th>Gr-news-BLEU</th>
<th>Gr-poetry-proposed</th>
<th>Gr-news-proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bing</td>
<td>56%</td>
<td>86%</td>
<td>67%</td>
<td>70%</td>
</tr>
<tr>
<td>Google</td>
<td>26%</td>
<td>8%</td>
<td>25%</td>
<td>16%</td>
</tr>
<tr>
<td>SDL</td>
<td>18%</td>
<td>6%</td>
<td>8%</td>
<td>14%</td>
</tr>
</tbody>
</table>

Table 4: Repeating the tests described in Table 1 with various pre-trained vector combinations.

<table>
<thead>
<tr>
<th>Proposed</th>
<th>En-En WMT’12</th>
<th>En-Gr WMT’17</th>
<th>Jp-Zh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google-WikiNeg</td>
<td>0.37</td>
<td>0.2</td>
<td>0.31</td>
</tr>
<tr>
<td>WikiNeg-WikiNeg</td>
<td>0.34</td>
<td>0.2</td>
<td>0.24</td>
</tr>
<tr>
<td>Google-WikiNorm</td>
<td>-0.01</td>
<td>0.05</td>
<td>-0.01</td>
</tr>
<tr>
<td>WikiNeg-WikiNorm</td>
<td>0.16</td>
<td>-0.02</td>
<td>0.07</td>
</tr>
</tbody>
</table>

4 Conclusion

In this work, we defined a new MT quality estimation setting where the proposed algorithm has no access whatsoever to the test text’s distribution or to an annotated training set with human judgment scores. In fact, we proposed an MT quality estimation system based on a new notion of sentence cohesiveness that we introduced. We tested our system on standard competition datasets for various language pairs. Our experimental results suggest that reasonable MT quality estimation can be carried out even in the restrictive setting.

Acknowledgments

The authors would like to express their sincere gratitude to the anonymous reviewers for their helpful and valuable comments. Also, the authors would like to thank Dr. Li Zhou (Brigham and Women’s Hospital - Harvard Medical School) for her unconditional support.

References


Domain Adaptation for Hindi-Telugu Machine Translation using Domain Specific Back Translation  

Hema Ala  
LTRC  
IIIT Hyderabad  
hema.alaresearch.iiit.ac.in  

Vandan Mujadia  
LTRC  
IIIT Hyderabad  
vandan.mu@research.iiit.ac.in  

Dipti Misra Sharma  
LTRC  
IIIT Hyderabad  
dipti@iiit.ac.in  

Abstract  

In this paper, we present a novel approach for domain adaptation in Neural Machine Translation which aims to improve the translation quality over a new domain. Adapting new domains is a highly challenging task for Neural Machine Translation on limited data, it becomes even more difficult for technical domains such as Chemistry and Artificial Intelligence due to specific terminology, etc. We propose Domain Specific Back Translation method which uses available monolingual data and generates synthetic data in a different way. This approach uses Out Of Domain words. The approach is very generic and can be applied to any language pair for any domain. We conduct our experiments on Chemistry and Artificial Intelligence domains for Hindi and Telugu in both directions. It has been observed that the usage of synthetic data created by the proposed algorithm improves the BLEU scores significantly.

1 Introduction  

Neural Machine Translation (NMT) systems achieved a breakthrough in translation quality recently, by learning an end-to-end system (Bahdanau et al., 2014)(Sutskever et al., 2014). These systems perform well on the general domain on which they trained, but they fail to produce good translations for a new domain the model is unaware of.  

Adapting to a new domain is highly challenging task for NMT systems, it becomes even more challenging when it comes to technical domains like Chemistry, Artificial Intelligence etc, as they contain many domain specific words. In a typical domain adaptation scenario like ours, we have a tremendous amount of general data on which we train an NMT model, we can assume this as a baseline model, now provided a new domain data, the challenge is to improve the translation quality of that domain using available little amount of parallel domain data. We adopted two technical domains namely, Chemistry and Artificial Intelligence for Hindi -> Telugu and Telugu -> Hindi experiments.

The parallel data for the mentioned technical domains is very less, hence we used back translation to create synthetic data. Instead of using synthetic data directly which may contain lots of noise we used domain monolingual data to create synthetic data in a different way (see section 3.4) and used such that translation of domain terms and context around them is accurate.

2 Background & Motivation  

As noted by Chu and Wang (2018) there are two important distinctions to make in domain adaptation methods for Machine Translation(MT). The first is based on data requirements, supervised adaptation relies on in-domain parallel data, and unsupervised adaptation has no such requirement. There is also a difference between model-based and data-based methods. Model-based methods make explicit modifications to the model architecture such as jointly learning domain discrimination and translation (Britz et al., 2017), interpolation of language modeling and translation (Gulcehre et al., 2015; Domhan and Hieber, 2017) and domain control by adding tags and word features (Kobus et al., 2016). Zeng et al. (2019) proposed iterative dual domain adaptation framework for NMT, which continuously fully exploits the mutual complementarity between in domain and out domain corpora for translation knowledge transfer. Apart from this Freitag and Al-Onaizan (2016) proposed two approaches,
one is to continue the training of the baseline model (general model) only on the in-domain data (domain data) and the other is to ensemble the continue model with the baseline model at decoding time. Coming to the data-based methods for domain adaptation, it can be done in two ways, combining in-domain and out-of-domain parallel corpora for supervised adaptation (Luong et al., 2015) or by generating pseudo-parallel corpora from in-domain monolingual data for unsupervised adaptation (Sennrich et al., 2015a; Currey et al., 2017).

Our approach follows a combination of both supervised and unsupervised approaches. where we first combine domain data (Chemistry and Artificial Intelligence) with general data, train a domain adaptation model. Then, as an unsupervised approach we use available domain monolingual data to back translate and use to create domain adaptation model. Burlot and Yvon (2019) explained how we can use monolingual data effectively in our MT systems, inspired from Burlot and Yvon (2019), instead of just adding domain parallel data which is very small in amount to general data we used available domain monolingual data to generate synthetic parallel data.

In Burlot and Yvon (2019) there have analyzed various ways to integrate monolingual data in an NMT framework, focusing on their impact on quality and domain adaptation. A simple way to use monolingual data in MT is to turn it into synthetic parallel data and let the training procedure run as usual (Bojar and Tammyna, 2011), but this kind of synthetic data may contain huge noise which leads to performance degradation of domain data. Therefore, we present an approach which generates synthetic data in a way such that it is more reliable and improves the translation. In the context of phrase-based statistical machine translation Daume III and Jagarlamudi (2011) has noted that unseen (OOV) words account for a large portion of translation errors when switching to new domains, however this problem is still exist even in NMT as well. Considering this issue, inspired from Huck et al. (2019) we proposed a novel approach called domain specific back translation which uses Out Of Domain (OOD) words to create synthetic data from monolingual data which will be discussed in detail in section 3.4. Huck et al. (2019) also created synthetic data using OOV in a different way, whereas we used OOD words to create synthetic data.

3 Methodology

As discussed in section 2 there are many approaches for domain adaptation mainly divided into model-based and data-based methods. However our approach falls under data-based method, we discuss this in detail in section 3.3. Though, there exists many domain adaptation works in MT, to the best of our knowledge there is no such work for Indian languages especially which considers technical domains like Chemistry, Artificial Intelligence etc. Hence there is a huge need to work on Indian languages where most of them are morphologically rich and these type of domains (technical domains) to improve the translation of domain specific text that contain many domain terms etc.

We conducted all our experiments for Hindi and Telugu in both directions for Chemistry and Artificial Intelligence. The language pair (Hindi-Telugu) considered in our experiments are morphologically rich therefore, there exists many post positions, inflections etc. In order to handle all these morphological inflections we used Byte Pair Encoding (BPE), we can see detail explanation about BPE in section 3.2.

3.1 Neural Machine Translation

NMT system attempts to find the conditional probability of the target sentence with the given source sentence. There exist several techniques to parameterize these conditional probabilities. Kalchbrenner and Blunsom (2013) used combination of a convolution neural network and a recurrent neural network, Sutskever et al. (2014) used a deep Long Short Term Memory (LSTM) model, Cho et al. (2014) used an architecture similar to the LSTM, and Bahdanau et al. (2014) used a more elaborate neural network architecture that uses an attention mechanism over the input sequence. However all these approaches are based on RNN’s and LSTM’s etc, but because of the characteristics of RNN, it is not conducive to training data in parallel so that
the model training time is often longer, by addressing this issue Vaswani et al. (2017) proposed Transformer framework based on a self-attention mechanism. Inspired from Vaswani et al. (2017) we used Transformer architecture in all our experiments.

3.2 Byte Pair Encoding
BPE (Gage, 1994) is a data compression technique that substitutes the most frequent pair of bytes in a sequence with a byte that does not occur within that data. Using this we can acquire the vocabulary of desired size and can handle rare and unknown words as well (Sennrich et al., 2015b). As Telugu and Hindi are morphologically rich languages, particularly Telugu being more Agglutinative language, there is a need to handle post positions and compound words, etc. BPE helps the same by separating suffix, prefix, and compound words. NMT with BPE made significant improvements in translation quality for low resource morphologically rich languages (Pinnis et al., 2017). We also adopted the same for our experiments and got the best results with a vocabulary size of 30000.

3.3 Domain Adaptation
Domain adaptation aims to improve the translation performance of a model (trained on general data) on a new domain by leveraging the available domain parallel data. As discussed in section 2 there are multiple approaches to do it broadly divided into model-based and data-based however, our approach falls under data-based methods, where one can combine the available little amount of domain parallel data to general data. In this paper we show how usage of domain specific synthetic data improves the translation performance significantly. The main goal of this method is to use domain-specific synthetic parallel data using the approach mentioned in section (3.4) along with little amount of domain parallel data.

3.4 Domain Specific Back Translation
In our experiments we followed data-based approach, we combined domain data with general data and trained a new model as a domain adaptation model.

Due to the fact that the domain data is very less we can use available monolingual data to

<table>
<thead>
<tr>
<th>Algorithm 1: Generic Algorithm for Domain Specific Back Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let us say $L_1$ and $L_2$ are language pair (translation can be done in both directions $L_1 \rightarrow L_2$ and $L_2 \rightarrow L_1$)</td>
</tr>
<tr>
<td>1. Training Corpus : Take all available $L_1$ - $L_2$ data (except domain data)</td>
</tr>
<tr>
<td>2. Train two NMT models (1. $L_1 \rightarrow L_2$ [L1-L2] 2. $L_2 \rightarrow L_1$ [L2-L1])</td>
</tr>
<tr>
<td>3. for domain in all domains do</td>
</tr>
<tr>
<td>1. Take $L_1$ domain data , list down all Out Of Domain words from $L_1$ Training Corpus [say this is OOD$L_1$ with respect to given domain]</td>
</tr>
<tr>
<td>2. Take $L_2$ domain data, list down all Out Of Domain words from $L_2$ Training Corpus [say this is OOD$L_2$ with respect to given domain]</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>4. Now take monolingual data for $L_1$ and $L_2$</td>
</tr>
<tr>
<td>5. for all domains do</td>
</tr>
<tr>
<td>1. Get N sentences from $L_1$ monolingual data where OOD$L_1$ are present [Mono-$L_1$]</td>
</tr>
<tr>
<td>2. Get N sentences from $L_2$ monolingual data where OOD$L_2$ are present [Mono-$L_2$]</td>
</tr>
<tr>
<td>3. Run $L_2$-$L_1$ on Mono-$L_2$ to get Back Translated data for $L_1 \rightarrow L_2$ (BT[L1-L2])</td>
</tr>
<tr>
<td>4. Run $L_1$-$L_2$ on Mono-$L_1$ to get Back Translated data for $L_2 \rightarrow L_1$ (BT[L2-L1])</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

* Steps to Extract OOD words(mentioned in step 3) for all domains for all languages: |
* for word in unique words of domain data do |
  * if word not in unique words of general data then that will be extracted as OOD word with respect to that domain |
end
generate synthetic parallel data. Leveraging monolingual data attained significant improvements in NMT (Domhan and Hieber, 2017; Burlot and Yvon, 2019; Bojar and Tamechna, 2011; Gulcehre et al., 2015). Using back translation we can generate synthetic parallel data but that might be very noisy which will decrease the domain specific translation performance. Hence we need an approach which extracts only useful sentences and creates synthetic data. Our approach addresses the same by creating domain specific back translated data using the algorithm mentioned in 1.

Domain-specific Back Translation tries to improve overall translation quality, particularly translation of domain terms and domain-specific context implicitly. The generic algorithm for domain-specific back translation is described in Algorithm 1. The algorithm is very generic and can be applied to any language pair for any domain. In our experiments, we adopted two domains namely Chemistry and Artificial Intelligence, one language pair Hindi and Telugu in both directions. Let us consider the mentioned languages in terms of algorithm mentioned in Algorithm 1 where L1 as Hindi and L2 as Telugu, domains are Chemistry and Artificial Intelligence. Now, each step of the algorithm can be interpreted as follows. step 1. The training corpus is general data mentioned in Table 1. step 2. We train 2 models using the training corpus from above step. One from Hindi to Telugu and the other is from Telugu to Hindi. These models can be treated as base models. step 3. This step is to find out OOD words, this can be done as follows, In Algorithm 1 this step explained in detail at the last. step 3.1 Get Unique words from general corpus, say Gen-Unique for both the languages step 3.2 Get Unique words from Chemistry corpus, Chem-Unique for both the languages step 3.3 Get Unique words from AI corpus, AI-Unique for both languages step 3.4 Now, take each word from Chem-Unique and check that word in Gen-Unique If it not found then that can be considered as Chemistry OOD words. We get OOD Hindi and OOD Telugu with respect to Chemistry. step 3.5 take each word from AI-Unique and check that word in Gen-Unique If it not found then that can be considered as AI OOD words. We get OOD words Hindi and OOD words Telugu with respect to AI. step 4. Take monolingual data for both languages mentioned in 2. step 5. Extract sentences from Hindi monolingual data where Hindi OOD words w.r.t Chemistry are present[Chem-Mono-Hindi]. step 5.1 Extract sentences from Telugu monolingual data where Telugu OOD words w.r.t Chemistry are present[Chem-Mono-Telugu]. step 5.2 Extract sentences from Hindi monolingual data where Hindi OOD words w.r.t AI are present[AI-Mono-Hindi]. step 5.3 Extract sentences from Telugu monolingual data where Telugu OOD words w.r.t AI are present[AI-Mono-Telugu]. step 6. Run Hindi -> Telugu model from step 2 on Chem-mono-Hindi to get Back Translated data [BT-Chem-Hindi-Telugu] step 6.1 Run Telugu -> Hindi model from step 2 on Chem-mono-Telugu to get Back Translated data [BT-Chem-Telugu-Hindi] step

<table>
<thead>
<tr>
<th>Domains</th>
<th>#Sentences</th>
<th>#Tkns(te)</th>
<th>#Unique Tkns(te)</th>
<th>#Tkns(hi)</th>
<th>#Unique Tkns(hi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>431975</td>
<td>5021240</td>
<td>443052</td>
<td>7995403</td>
<td>123716</td>
</tr>
<tr>
<td>AI</td>
<td>5272</td>
<td>57051</td>
<td>11900</td>
<td>89392</td>
<td>5479</td>
</tr>
<tr>
<td>Chemistry</td>
<td>3600</td>
<td>72166</td>
<td>10166</td>
<td>97243</td>
<td>6792</td>
</tr>
</tbody>
</table>

Table 1: Parallel Data for Hindi - Telugu

<table>
<thead>
<tr>
<th>Langs</th>
<th>#Sent</th>
<th>#Tkns</th>
<th>UTkns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hindi</td>
<td>16345</td>
<td>175931</td>
<td>17405</td>
</tr>
<tr>
<td>Telugu</td>
<td>39583</td>
<td>339612</td>
<td>86942</td>
</tr>
</tbody>
</table>

Table 2: Monolingual Data

<table>
<thead>
<tr>
<th>Domain-Lang</th>
<th>#Sentences</th>
<th>#Tkns</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI-Hindi</td>
<td>14014</td>
<td>43848</td>
</tr>
<tr>
<td>AI-Telugu</td>
<td>22241</td>
<td>285234</td>
</tr>
<tr>
<td>Chemistry-Hindi</td>
<td>28672</td>
<td>982700</td>
</tr>
<tr>
<td>Chemistry-Telugu</td>
<td>34322</td>
<td>425515</td>
</tr>
</tbody>
</table>

Table 3: Selected monolingual data for domain specific back translation
6.2 Run Hindi -> Telugu model from step 2 on AI-mono-Hindi to get Back Translated data [BT-AI-Hindi-Telugu] step 6.3 Run Telugu -> Hindi model from step 2 on AI-mono-Telugu to get Back Translated data [BT-AI-Telugu-Hindi].

The data we get from step 6 is the one produced by our proposed algorithm. This domain specific synthetic data can be used to improve the domain adaptation performance. The way of extracting sentences where OOD words are present ensure that we only select sentences where domain terms/domain specific terms were present instead of all sentences which may produce lots of noise.

In our experiments we compare four models in each domain, the general model is common for both the domains. The first model is general or baseline model which trains on only general data, then we add very less amount of parallel domain data for each domain separately which is called domain adaptation model. Then comes our domain specific synthetic data, we combine this in two ways. The third model is adding domain specific synthetic data to the general data and the fourth model is the proposed one which adds domain specific back translated data to the training data used for basic domain adaptation model(general+domain, model2). Therefore we have seven models in total, one is general and three models for Chemistry and AI independently.

4 Experimental Setup

4.1 Data

Based on the above mentioned approaches, we carried out our experiments on datasets mentioned in Tables 1, 2, 3 of parallel data, monolingual data, selected monolingual data for domain specific back translation respectively. we got general parallel data from OPUS corpus (Tiedemann, 2012) and the ILCI (Indian Languages Corpora Initiative) corpus (Jha, 2010), similarly domain data from ICON Adap-MT 2020 shared task 1 for Chemistry and AI. We can see statistics of overlapping tokens and Out of Vocabulary (OOV) tokens (we can assume these as Out Of Domain words) in Tables 4, 5.

We extracted Chemistry and AI monolingual data from Wikipedia and combined for respective language(see Table 2). In absence of domain monolingual data for any domains one can do same experiments with general monolingual data as well. As Hindi and Telugu are morphologically rich languages, Telugu being more inflected language we apply approximate matching to allow for some morphological variations in the terms for mining the sentences from monolingual data using OOD words we got for respective languages and respective domains (see Table 3). If we observe Table 2 and 3 the Chemistry-Hindi got 28672 sentences but the actual monolingual data for Hindi is 16345 which is less than selected monolingual data. This happened because when we have two OOD words present in a sentence then we select that sentence two times for each word one time.

<table>
<thead>
<tr>
<th>Domain-Language</th>
<th>#Overlapped Tkns</th>
<th>#OOV Tkns</th>
</tr>
</thead>
<tbody>
<tr>
<td>General-Hindi</td>
<td>3777</td>
<td>1702</td>
</tr>
<tr>
<td>General-Telugu</td>
<td>6164</td>
<td>5736</td>
</tr>
<tr>
<td>Chemistry-Hindi</td>
<td>1931</td>
<td>3548</td>
</tr>
<tr>
<td>Chemistry-Telugu</td>
<td>2036</td>
<td>9864</td>
</tr>
<tr>
<td>monolingual-Hindi</td>
<td>1977</td>
<td>3502</td>
</tr>
<tr>
<td>monolingual-Telugu</td>
<td>3487</td>
<td>8413</td>
</tr>
</tbody>
</table>

Table 4: Vocab overlap across domains for AI for respective language

<table>
<thead>
<tr>
<th>Domain-Language</th>
<th>#Overlapped Tkns</th>
<th>#OOV Tkns</th>
</tr>
</thead>
<tbody>
<tr>
<td>General-Hindi</td>
<td>4038</td>
<td>2754</td>
</tr>
<tr>
<td>General-Telugu</td>
<td>5773</td>
<td>4393</td>
</tr>
<tr>
<td>AI-Hindi</td>
<td>1381</td>
<td>4864</td>
</tr>
<tr>
<td>AI-Telugu</td>
<td>2036</td>
<td>8130</td>
</tr>
<tr>
<td>monolingual-Hindi</td>
<td>2587</td>
<td>4205</td>
</tr>
<tr>
<td>monolingual-Telugu</td>
<td>3967</td>
<td>6199</td>
</tr>
</tbody>
</table>

Table 5: Vocab overlap across domains for Chemistry for respective language

4.2 Training Details

We used OpenNMT-py toolkit (Klein et al., 2018) for all our experiments. We used Transformer model with 6 layers in both encoder and decoder each with 512 hidden units. The word embedding size is set to 512 with 8 heads and dropout is set to 0.3 to avoid over fitting of the model. We used perplexity as an early stopping criteria.
5 Results & Discussion

When we say back chem or back AI in this paper, that means they are the domain specific back translated data. After getting domain specific synthetic data from algorithm 1, we combined that in two ways and trained models. One is directly adding the domain specific back translated data to general parallel data without any actual domain data, second is to add domain specific back translated to the general parallel data along with actual domain parallel data.

We evaluate our models on test data set provided by ICON Adap-MT 2020 shared task using widely used automatic MT evaluation metric called BLEU (Papineni et al., 2002). The models presented in 6 are trained based on data-based approach only. However, the model which used the domain specific synthetic parallel data along with actual domain parallel data outperformed other models.

The first model is M1-Gen which is trained on only general parallel data this is same for both the domains(Chemistry and Artificial Intelligence) it performed well on general data but not on chemistry and AI data for both directions Telugu->Hindi and Hindi->Telugu. Then the second model is a domain adaption model where we combine the available little amount of domain parallel data with general data(M2-gen+chem and M2-gen+AI). These models outperformed the general model on Chemistry(9.5 to 11.8) and AI(8.7 to 10.3) domains though there is a decrease of BLEU score on general data for Telugu->Hindi, this pattern is same for Hindi->Telugu as well. Third model is adding domain specific synthetic data directly to the general data(M3-Gen+Back Chem and M3-Gen+Back AI) which decreased the BLEU score compared to domain adaptation model(M2 for both the domains) but it’s improved little bit from general model. For Chemistry domain in Telugu->Hindi the BLEU score decreased from 11.8 to 9.9, but it increased from general model(9.5 to 9.9). Now the fourth model is the proposed approach which adds domain specific back translated data to the data used in initial domain adaptation model(gen+domain data) for both the domains(M4-Gen+Chem+back Chem and M4-Gen+AI+back AI). M4 model in both domains outperformed all other models. There is an increase 3.4 BLEU points in Chemistry for Telugu->Hindi from general model to proposed model which uses domain specific synthetic data.

The thought behind the domain-specific back translation is selecting sentences where the unseen words (most of them are domain terms) are present. By doing this, the model implicitly learns the translation of domain terms and context around them accurately.
address this point we present two examples below which shows an overall improvement of domain specific text translation including domain terms.

Table 7 represents an example from Telugu to Hindi NMT system for Chemistry domain. This example contains a domain specific term రైబోనూయ్కుల్యేజ్ (raibōnyūkkyēj) which was wrongly translated by other models except our model which uses domain specific back translated text (Gen+Chem+back Chem). If we consider another example in Table 8 from Telugu to Hindi model, the same pattern is observed as above. The proposed approach translated the domain term (sentriphyoogeshan) and overall sentence correctly whereas others failed to do it. From these examples we can observe our proposed approach is handling domain specific terms implicitly and translating them better compared to others. As we are mining sentences from monolingual data where the out of vocabulary words with respect to each domain (we can treat them as domain terms which are not present in general data) are present, by doing this it ensure to translate unseen words especially domain terms properly.

In Table 6, the BLEU score of AI is improved with the gen+Chem model compared to the gen model, the same pattern is observed for Chemistry as well. From this, we can assume there is a similarity between these domains in terms of domain terms or context, etc. Based on this assumption we can further experiment models with combination of similar domains.

6 Conclusion and Future work

We presented an approach called domain specific back translation to produce synthetic data from available monolingual data which can be applied to any language pair for any domain. we did our experiments on two domains Chemistry and Artificial Intelligence for Hindi and Telugu. The approach follows extracting Out Of Domain words from large amount of general data with respect to particular domain (here Chemistry and AI), then mining the sentences from domain monolingual data where these OOD words are present. By doing this the system will learn to translate unknown words and domain terms properly. Without adding direct monolingual data which contains lots of noise, we select only sentences where general OOD words with respect to a domain are present. In this paper we showed how addition of domain specific back translated data to the general and little amount of domain data improved the translation performance in terms of BLEU scores. From the results it has been observed that the proposed approach improving BLEU score significantly. we would like to apply this generic approach to all possible Indian languages and multiple domains with combination of similar domains.

References


<table>
<thead>
<tr>
<th>Type/Model</th>
<th>Sentence</th>
<th>Source (Telugu source sentence)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source</strong></td>
<td>కావున, మనం రైబోనూయ్కిల్ యేజ్ ను రెండు విభినన్ మారాగ్ లలో రీఫోల్డ్ చేయవచుచు.</td>
<td>కావున, మనం రైబోనూయ్కిల్ యేజ్ ను రెండు విభినన్ మారాగ్ లలో రీఫోల్డ్ చేయవచుచు. (Kavuna, manam raibonyükliyey nu reŋdu vibhinna mąrgalph répholč ceyavaccu.)</td>
</tr>
</tbody>
</table>

| **Target** | హమ రాయబాయియింలియాండ్ కో దే అలగ - అలగ తరికిలో సెరిఫోఎల్ కరి సకతేంటాం | (ham raibonyookleyez ko do alag - alag tareekon se reephold kar sakate hain.) |

| **Model1** | (model trained on only general data) | ఇస్ కారణు, హమ రాయబాయియింలియాండ్ కో దే అలగ - అలగ తరికిలో సెరిఫోఎల్ కరి సకతేంటాం | (is kaaran, raibonyookleyez ko do vibhinna tareekon se riphon kar sakate hain.) |

| **Model2** | (model trained on general+Chemistry data) | తి, హమ రాయబాయియింలియాండ్ కో దే అలగ - అలగ తరికిలో సెరిఫోఎల్ కరి సకతేంటాం | (to, ham raabonyookliyotaid ko do alag tareekon se reephold kar sakate hain.) |

| **Model3** | (model trained on general+ back translated Chemistry data) | అత: హమ రాయబాయియింలియాండ్ కో దే అలగ - అలగ తరికిలో సెరిఫోల్డ్ కరి సకతేంటాం | (at: ham raibonyookleyyar ko do vibhinna tareekon se reephold kar sakate hain.) |

| **Model4** (Proposed Model: trained on general + Chemistry + back translated Chemistry data) | ఇసిపింది, హమ రాయబాయియింలియాండ్ కో దే అలగ - అలగ తరికిలో సెరిఫోల్డ్ కరి సకతేంటాం | (isalie, ham raibonyookleyez ko do alag - alag tareekon se reephold kar sakate hain.) |

| **Table 7**: Telugu -> Hindi Example from improved sentences for Chemistry domain | |

<table>
<thead>
<tr>
<th>Type/Model</th>
<th>Sentence</th>
<th>Source (Telugu source sentence)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source</strong></td>
<td>సెంటర్ ఫూగేషన్ మరియు ఉంది</td>
<td>(sentriphyoogeshan aur usake siddhaant.)</td>
</tr>
</tbody>
</table>

| **Target** | సెంటర్ ఫూయ్గేషన్ మరియు ఉంది | (Seṇṭriphyūgēṣan mariyu dāni sūtraṁ.) |

| **Model1** | (model trained on only general data) | సెంటర్ ఫూయ్గేషన్ మరియు ఉంది | (Seṇṭriphyūgēṣan mariyu dāni sūtraṁ.) |

| **Model2** | (model trained on general+Chemistry data) | సెంటర్ ఫూయ్గేషన్ మరియు ఉంది | (Seṇṭriphyūgēṣan, dāni suṭrālu.) |

| **Model3** | (model trained on general+ back translated Chemistry data) | సెంటర్ ఫూయ్గేషన్ మరియు ఉంది | (Seṇṭriphyūṣan dāni suṭrāmu.) |

| **Model4** (Proposed Model: trained on general + Chemistry + back translated Chemistry data) | సెంటర్ ఫూయ్గేషన్ మరియు ఉంది | (Seṇṭriphyūgēṣan mariyu dāni sūtraṁ.) |

| **Table 8**: Hindi -> Telugu example from improved sentences for Chemistry domain | |


Girish Nath Jha. 2010. The tdil program and the indian language corpora intitiative (ilci). In LREC.


ArabGlossBERT: Fine-Tuning BERT on Context-Gloss Pairs for WSD

Moustafa Al-Hajj
Lebanese University
Beirut, Lebanon
moustafa.alhajj@ul.edu.lb

Mustafa Jarrar*
Birzeit University
Birzeit, Palestine
mjarrar@birzeit.edu

Abstract

Using pre-trained transformer models such as BERT has proven to be effective in many NLP tasks. This paper presents our work to fine-tune BERT models for Arabic Word Sense Disambiguation (WSD). We treated the WSD task as a sentence-pair binary classification task. First, we constructed a dataset of labeled Arabic context-gloss pairs (~167k pairs) we extracted from the Arabic Ontology and the large lexicographic database available at Birzeit University. Each pair was labeled as True or False and target words in each context were identified and annotated. Second, we used this dataset for fine-tuning three pre-trained Arabic BERT models. Third, we experimented the use of different supervised signals used to emphasize target words in context. Our experiments achieved promising results (accuracy of 84%) although we used a large set of senses in the experiment.

1 Introduction

Word Sense Disambiguation (WSD) aims to determine which sense (i.e. meaning) a word may denote in a given context. This is a challenging task due to the semantic ambiguity of words. For example, the word “book” as a noun has ten different senses in Princeton WordNet such as “a written work or composition that has been published” and “a number of pages bound together”. WSD has been a challenging task for many years but has gained recent attention due to the advances in contextualized word embedding models such as BERT (Devlin et al., 2019), ELMo (Peters et al., 2018) and GPT-2 (Radford et al., 2018). Such language models require less labeled training data since they are initially pre-trained on large corpora using self-supervised learning. The pre-trained language models can then be fine-tuned on various downstream NLP tasks such as sentiment analysis, social media mining, Named-Entity Recognition, word sense disambiguation, topic classification and summarization, among others.

A gloss is a short dictionary definition describing one sense of a lemma or lexical entry (Jarrar, 2006, 2005). A context is an example sentence in which the lemma or one of its inflections (i.e. the target word) appears. In this paper, we aim to fine-tune Arabic models for Arabic WSD. Given a target word in a context and a set of glosses, we will fine-tune BERT models to decide which gloss is the correct sense of the target word. To do that, we converted the WSD task into a BERT sentence-pair binary classification task similar to (Huang et al., 2019; Yap et al., 2020; Blevins and Zettlemoyer, 2020). Thus, BERT is fine-tuned on a set of context-gloss pairs, where each pair is labeled as True or False to specify whether or not the gloss is the sense of the target word. In this way, the WSD task is converted into a sentence-pair classification task.

One of the main challenges for fine-tuning BERT for Arabic WSD is that Arabic is a low-resourced language and that there are no proper labeled context-gloss datasets available.

To overcome this challenge, we collected a relatively large set of definitions from the Arabic Ontology (Jarrar, 2021) and multiple Arabic dictionaries available at Birzeit University’s lexicographic database (Jarrar and Amayreh, 2019; Jarrar et al., 2019) then we extracted glosses and contexts from lexicicon definitions.

Another challenge was to identify, locate and tag target words in context. Tagging target words with special markers is important in the fine-tuning phase because they act as supervised signals to highlight these words in their contexts, as will be explained in section 5. Identifying target words is not straightforward as they are typically inflections of lemmas, i.e. with different spellings. Moreover, locating them is another challenge as the same word may appear multiple times in the
same context with different senses. For example, the word (تَنْبًى) appears two times in this context with two different meanings: went and gold. We used several heuristics and techniques (as described in subsection 3.3) to identify and locate target words in context in order to tag them with special markers.

As a result, the dataset we constructed consists of about 167K context-gloss pair instances, 60K labeled as True and 107K labeled as False. The dataset covers about 26k unique lemmas (undiacritized), 32K glosses and 60k contexts.

We used this dataset to fine-tune three pre-trained Arabic BERT models: AraBERT (Antoun et al., 2020), QARiB (Abdelali et al., 2021) and CAmeLBERT (Inoue et al., 2021)

Each of the three models was fine-tuned for context-gloss binary classification. Furthermore, we investigated the use of different supervised signals used to highlight target words in context-gloss pairs.

The contributions of this paper can be summarized as follows:

1. Constructing a dataset of labeled Arabic context-gloss pairs;
2. Identifying, locating and tagging target words;
3. Fine-tuning three BERT models for Arabic context-gloss pairs binary classification;
4. Investigating the use of different markers to highlight target words in context.

The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 describes the constructed dataset and our methodology to extract and label context-gloss pairs, and splitting the dataset into training and testing sets. Section 4 outlines the task we resolved in this paper and Section 5 presents the fine-tuning methodology. The experiments and the obtained results are presented in Sections 6 and 7 respectively. Finally, Section 8 presents conclusions and future work.

2 Related Work

Recent experiments in fine-tuning pre-trained language models for WSD and related tasks have shown promising results, especially those that use context-gloss pairs in fine-tuning such as (Huang et al., 2019; Yap et al., 2020; Blevins and Zettlemoyer, 2020).

Huang et al. (2019) proposed to fine-tune BERT on context-gloss pairs (label ∈ {yes, no}) for WSD, such that the gloss corresponding to the context-gloss pair candidate, with the highest output score for yes, is selected. Yap et al. (2020) proposed to group context-gloss pairs with the same context but different candidate glosses as 1 training instance (groups of 4 and 6 instances). Then, they proposed to fine-tune BERT model on group instances with 1 neuron in the output layer. After that, they formulated WSD as a ranking/selection problem where the most probable sense is ranked first.

Others also suggested to emphasize target words in context-gloss training instances. Huang et al. (2019); Botha et al. (2020); Lei et al. (2017); Yap et al. (2020) proposed to use different special signals in the training instance, which makes the target word “special” in it. As such, Huang et al. (2019) proposed to use quotation marks around target words in context. In addition, they proposed to add the target word followed by a colon at the beginning of each gloss, which contributes to emphasizing the target word in the training instance. Yap et al. (2020) proposed to surround the target word in context with two special [TGT] tokens. In contrast, Botha et al. (2020); Lei et al. (2017) proposed to surround the target word in context with two different special tokens as marks of opening and closing. In this paper, we investigate the use of different types of signals to emphasize target words in context for Arabic WSD.

El-Razzaz et al. (2021) fine-tuned two BERT models on a small dataset of context-gloss pairs, consisting of about 5k lemmas, about 15k positive and 15k negative context-gloss pairs. They claimed an F1-score of 89%. However, this result is not reliable. After repeating the same experiment, we found that the majority of the context sentences used in the tests were already used for training. In this paper, we carefully selected the test set such that no contexts are used in both the training and the test sets. Additionally, we used a much larger sense repository (26k lemmas, 33k concepts and 167k context-gloss positive and negative pairs), which makes the task more challenging.

Other works related to Arabic WSD includes the use of static embeddings such as context and

3 Dataset Construction

This section describes how we constructed a dataset of labeled Arabic context-gloss pairs (See examples of pairs in Figure 1). We extracted the context-gloss pairs from the Arabic Ontology and multiple lexicons in the Birzeit University’s lexicographic database. The extracted pairs are labeled as True, and based on these True pairs, we generated the False pairs. Additionally, we identified the target word in each context and tagged it with different types of markers.

3.1 Context-Gloss Pairs Extraction

Arabic is a low-resource language (Darwish et al., 2021) and there are no proper sense repositories available for Arabic (Naser-Karajah et al., 2021; Jarrar et al., 2021) that can be used to generate a dataset of context-gloss pairs, e.g. similar to the Princeton WordNet for English (Miller et al., 1990). The largest available lexical-semantic resource for Arabic is the Birzeit University’s lexicographic database, which contains the Arabic Ontology (Jarrar, 2021, 2011) and about 400K glosses extracted from about 150 lexicons (Jarrar and Amayreh, 2019; Jarrar et al., 2019; Alhafi et al., 2019). The problem is that each of the 150 lexicons covers a partial set of glosses and lemmas. Thus, for a given lemma, collecting the glosses from all lexicons may result in a set of redundant senses. Another problem is that some lexicons provide multiple senses within the same definition with no clear structure or separation markers, which makes it difficult to extract senses. Furthermore, some lexicons do not provide contexts (i.e. example sentences) or they mix them with the definitions.

To overcome the above challenges and build a context-gloss pairs dataset, we performed the following steps:

**First, selection of candidate definitions:** We acquired the 400K lexicon definitions to select a set of good candidate definitions. A good definition represents either one sense or multiple senses that are easy to parse and split (i.e. contains some markers) and has context examples. That is, definitions that are not easy to parse or do not provide contexts were excluded.

**Second, extraction of glosses and contexts:** Each of the collected candidate definitions in the first phase was parsed and split into gloss(es) and context(s). Some definitions did not need to be split and some were split into separate glosses (one for each sense) in case a definition contains multiple glosses (i.e. senses). Contexts were also extracted from the candidate definitions, taking into account that a definition may include multiple contexts for one sense. A parser was developed for each lexicon as each lexicon has its structure and text markers. Nevertheless, some lexicons were clean and well-structured (e.g. the Arabic Ontology) that did not need any parsing.

**Third, selection of glosses and contexts:** Given that the glosses and contexts were extracted in the second phase, we applied the following criteria to select the glosses and contexts that we need to build a dataset of context-gloss pairs:

- Short glosses and contexts (i.e. one-word long) were excluded as they do not add useful information in the fine-tuning phase.
- For each lemma, if one of its glosses does not have a context example then all glosses for this lemma were not selected. That is, for a lemma and its glosses to be selected, each gloss must have at least one context example.
- In case the same lemma appears in multiple lexicons, the one with more glosses was selected. For example, let \( m \) be a lemma with two glosses in lexicon A and three glosses in lexicon B, then the lexicon B set of glosses for \( m \) is favored. If the same lemma has an equal number of glosses in multiple lexicons, we manually favor the more renowned lexicon. The idea of favoring lemmas with more glosses is because it indicates a richer set of distinct senses, and in this way, we avoid redundant senses for the same lemma in the dataset.

---

2Lexicographic Search Engine: https://ontology.birzeit.edu/about

3We used the same parsing framework developed by (Amayreh et al., 2019) for lexicon digitization.
Table 1: Statistics about our context-gloss pairs dataset

<table>
<thead>
<tr>
<th></th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Lemmas (undiacritized)</td>
<td>26169</td>
</tr>
<tr>
<td>Avg glosses per Lemmas</td>
<td>1.25</td>
</tr>
<tr>
<td>Unique Glosses</td>
<td>32839</td>
</tr>
<tr>
<td>Unique Contexts</td>
<td>60272</td>
</tr>
<tr>
<td>Avg context per gloss</td>
<td>1.83</td>
</tr>
<tr>
<td>True context-gloss pairs</td>
<td>60323</td>
</tr>
<tr>
<td>False context-gloss pairs</td>
<td>106884</td>
</tr>
<tr>
<td>Total True and False pairs</td>
<td>167207</td>
</tr>
</tbody>
</table>

- Only glosses for single-word lemmas are selected. Although multi-word expression lemmas are important, in this phase, we only focus on single-word lemmas as BERT can process single-word tokens. We plan to consider multi-word lemmas in the future.

As a result, we selected about 32k glosses and 60k contexts for about 26K single lemmas (undiacritized), resulting in about 60k context-gloss pairs that we labeled as True pairs (see Table 1 for more statistics). It is important to note that our dataset cannot be considered an Arabic sense repository because a sense repository should contain all senses for a given lemma, but our dataset does not necessarily include all senses for every lemma.

### 3.2 Labeling Context-Gloss Pairs

The 60k context-gloss pairs extracted in the previous phase were labeled as True. The False context-gloss pairs were then generated based on the True pairs, as follows: For each lemma with more than one gloss, we cross-related its glosses with its contexts. For example, let (context1 - gloss1) and (context2 - gloss2) be the two True pairs for the same lemma, then (context1 - gloss1) and (context2 - gloss1) are generated and labeled as False pairs. As a result, about 107K context-gloss False pairs were generated in this way.

### 3.3 Annotating Target Words

This section presents our methodology for identifying the target word inside a given context and tagging it with a special supervised signal, which we need in the fine-tuning phase (see section 5). Figure 2 illustrates different tags of target words.

Given a lemma and a context, our goal is to identify which word is the target word in this context. As explained in section 1, a context is an example sentence in which a word (called target word) is mentioned with its sense defined in the gloss. Identifying a target word inside its context is not straightforward because: (i) it does not necessarily share the same spelling with its lemma, e.g., the word (محمود) and its lemma (محمد) and, more importantly, (ii) it might occur multiple times and each time with a different sense such as (كتب) which appears two times in this context (كتب عدة كتب), with two different meanings: wrote and books.

The following four methods were performed at the same time to maximize the certainty in identifying target words. The resulting target words were verified manually:

- **Sub-string**: We compared every word in the context with the given lemma (string-matching, after undiacritization). If the lemma is a sub-string of one or more words, then these words are candidate target words.

- **Character-level cosine similarity**: We developed a function\(^4\) that takes a lemma and a context and returns the word with the max cosine similarity with the lemma. The minimum cosine value should be more than 0.75 — an empirical threshold that we learned while reviewing the results. If a word is returned, then we considered it a candidate target word.

- **Levenshtein distance**: This function takes a lemma and a context and returns the word with max Levenshtein distance (after removing diacritics) by comparing each word in the context with the lemma. The returned word is considered a candidate target word.

- **Lemmatization**: We used our in-house lemmatizer and lexicographic database to lemmatize every word in the given context and return those words that have their lemmas the same as the given lemma. The returned words are considered candidate target words.

These four methods were applied in parallel to maximize the certainty of correct matching and identification of target words. The results (candidate words, their scores and position) of the four methods were then combined and sorted (from

\(^4\)The function converts two Arabic words (after removing diacritics) into two vectors (each cell represents the occurrence of a character), then computes their cosine similarity.
more to less certain) and given to linguists to review. Each identified target word\(^5\) was manually verified and, if needed, corrected by a linguist.

### 3.4 Training and Test Datasets

This section describes how we divided our dataset into training and test sets and the criteria we used to avoid repeated context in training and test sets. Recall that our dataset contains one or more glosses for each lemma and one or more contexts for each gloss, which we used to generate the context-gloss pairs dataset. The dataset cannot be arbitrarily divided as contexts used for training should not be used for testing. We selected the test set taking into account these two criteria: (i) every context selected in the test set should not be selected in the training set and (ii) every gloss should be selected in both the training and the test sets.

Given these criteria, we selected the test set as follows: (First) we selected the pairs with repeated glosses from the set of context-gloss pairs (i.e. glosses with more than one context). (Second) we grouped pairs according to their glosses then selected one pair from each group larger than one and included it in the test set. All of these pairs were labeled as True. (Third) we cross-related contexts with glosses of the same lemma to generate False pairs in the test set from the True pairs — as described in subsection 3.2. That is, again, the False pairs were generated after selecting the True pairs, and every pair selected for testing should not be part of the training set.

The resulted training and test datasets\(^6\) consist of 152,035 and 15,172 pairs, respectively. Table 2 provides statistics about the training and test sets.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Pairs</th>
<th>Count</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>True pairs</td>
<td>55,585</td>
<td></td>
</tr>
<tr>
<td></td>
<td>False pairs</td>
<td>96,450</td>
<td>152,035</td>
</tr>
<tr>
<td>Test</td>
<td>True pairs</td>
<td>4,738</td>
<td></td>
</tr>
<tr>
<td></td>
<td>False pairs</td>
<td>10,434</td>
<td>15,172</td>
</tr>
<tr>
<td></td>
<td><strong>Total</strong></td>
<td>60,323</td>
<td>167,207</td>
</tr>
</tbody>
</table>

Table 2: Counts of the training and testing pairs

### 4 Task Overview

Given a context, a target word in the context and a gloss, our task is to decide whether or not the gloss corresponds to a specific sense of the target word. We approached the problem as a binary sequence-pair classification task. We concatenated the context and the gloss and separated them by the special [SEP] token (See Figure 1). Afterward, we fine-tuned Arabic BERT models on our labeled dataset of context-gloss pairs \((label \in \{True, False\})\).

It is worth noting that although this binary context-gloss pair classification task is related to the WSD task, they are not exactly the same task. The WSD task aims at determining which sense (or gloss) a word in context denotes from a given set of senses. It is also worth noting that these two tasks are not the same as the Word-In-Context (WIC) task (Al-Hajj and Jarrar, 2021; Martelli et al., 2021), which aims at determining whether a target word has the same sense in two given contexts.

### 5 Methodology

To address the binary context-gloss classification task, we experimented with four variations of the context-gloss pairs. The idea is to investigate using different supervised signals around target words to give them special attention during the fine-tuning. Figure 2 illustrates these four variations. In variation 1, context-gloss pairs were left intact, without any signal. In the other three
Figure 2: Illustration of the four context-gloss pairs variations.

variations, we followed the techniques used by Huang et al. (2019), Yap et al. (2020) and Blevins and Zettlemoyer (2020) to signal target words. We surrounded target words with (i) single quotes in variation 2, (ii) the special token [UNUSED0] in variation 3, and (iii) [UNUSED0] before and after the target word in variation 4. Moreover, in the last three variations, we added the target word followed by a colon at the beginning of each gloss. In these four variations, the context and the gloss were concatenated into a sequence separated with the [SEP] token.

We fine-tuned three Arabic pre-trained models: AraBERT (Antoun et al., 2020), QARiB (Abdelali et al., 2021) and CAMeLBERT (Inoue et al., 2021) using our training dataset described in Section 3. Before fine-tuning AraBERT, we used the pre-processing method used in (Antoun et al., 2020) to pre-train version 2 of their model. Before fine-tuning CAMeLBERT and QARiB models, we used the pre-processing method used in (Inoue et al., 2021) to pre-train the CAMeLBERT which consists in the normalization of alif maksura (ạ), teh marbuta (ِ), alif (ا) and undiacritization.

Since BERT has a max length limit of tokens equal to 512, we limit the length of each training instance (i.e. context-gloss pair) with a maximum of 512 tokens. Given, for example, the tokenizer used in AraBERTv02, only 216 pairs are larger than 512 tokens out of the 167,207 pairs in our dataset. Instances shorter than 512 were padded to the max length limit.

The BertForSequenceClassification model architecture is used in fine-tuning the three Arabic BERT models. The last hidden state of the token [CLS] is used for the classification task. The linear layer in the output consists of two neurons for the True and False classes.

6 Experiment Setup

We selected the base configuration of AraBERTv02, QARiB, and CAMeLBERT models due to computational constraints and as larger models do not necessitate better performance (Abdelali et al., 2021; Inoue et al., 2021). We used the huggingface “Trainer” class in the fine-tuning. We performed a limited grid search to find a good hyperparameters combination then we fine-tuned each of the three models using the optimal configuration: initial learning rate of 2e-5, warmup_steps of 1412 with a batch size of 16 over 4 training epochs. All other hyperparameters were kept at their default values. We used a single Tesla P100-PCIE-16GB in fine-tuning models.

7 Results and Discussion

This section presents the results of two experiments. Table 3 presents the results of the first experiment in which we fine-tuned three BERT models on the variation 2 (i.e. single quotes signal) of context-gloss pairs.

As AraBERTv02 outperformed other models in the first experiment, it has been chosen for conducting a second experiment in which we fine-tuned on variation 1 (intact context-gloss pairs), variation 3 (two [UNUSED0] tokens around the target word in context-gloss pairs) and variation 4 ([UNUSED0] and [UNUSED1] tokens around the target word in context-gloss pairs). Reported results in Table 4 reveal that the use of different supervised signals around the target word did not significantly improve the overall results. The use of supervised signals reveals only 1% of improvement over variation 1 (no signals). This improvement is comparable to the improvement of 1-2% achieved by Huang et al. (2019) using special signals on English datasets.

8 Conclusion and Future Work

We presented a large dataset of context-gloss pairs (167,207 pairs) that we carefully extracted from the Arabic Ontology and diverse lexicon definitions. Each pair was labeled as True and False and each target word in each context was annotated and tagged. We used this dataset to fine-tune three Arabic BERT models on binary context-gloss
### Model Accuracy

<table>
<thead>
<tr>
<th>Model</th>
<th>True</th>
<th>False</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>AraBERTv02</td>
<td>81</td>
<td>85</td>
<td>84</td>
</tr>
<tr>
<td>CAMeLBERT</td>
<td>77</td>
<td>83</td>
<td>82</td>
</tr>
<tr>
<td>QARiB</td>
<td>73</td>
<td>82</td>
<td>80</td>
</tr>
</tbody>
</table>

Table 3: Achieved results (%) after fine-tuning three Arabic BERT models with the *single quotes* supervised signal around the target word.

<table>
<thead>
<tr>
<th>Variation</th>
<th>True</th>
<th>False</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variation 1</td>
<td>80</td>
<td>85</td>
<td>83</td>
</tr>
<tr>
<td>Variation 3</td>
<td>81</td>
<td>85</td>
<td>84</td>
</tr>
<tr>
<td>Variation 4</td>
<td>81</td>
<td>85</td>
<td>84</td>
</tr>
</tbody>
</table>

Table 4: Achieved results (%) with AraBERTv02 using the other three supervised signals around the target word.

We achieved a promising accuracy of 84%, especially as we used a large set of senses. Our experiments show that the use of different supervised signals around target words did not bring significant improvements (about 1%).

We will further build a large-scale content-gloss dataset. We also plan to include contexts written in Arabic dialects (Jarrar et al.) so that dialectal text can be sense-disambiguated. Additionally, we plan to consider Arabic text that is partially or fully diacritized, which requires lemmas across lexicons to be linked with each other (Jarrar et al., 2018). Lastly but more importantly, we plan to extend our work to address the WSD task and build a semantic analyzer for Arabic.

**Acknowledgments**

We would like to thank the reviewers for their valuable comments and efforts for improving our manuscript. We would also like to thank Taymaa Hammouda for her technical support while preparing the dataset and annotating the contexts. We extend our thanks to Dr Abeer Naser Eddine for proofreading this paper.

**References**


Rim Laatar, Chafik Alolou, and Lamia Hadrich Bellguith. 2017. Word sense disambiguation of arabic language with word embeddings as part of the creation of a historical dictionary. In LPKM.


Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke


English-Arabic Cross-language Plagiarism Detection

Naif Alotaibi
Department of Computer Science
University of Warwick
naif.alotaibi@warwick.ac.uk

Mike Joy
Department of Computer Science
University of Warwick
m.s.joy@warwick.ac.uk

Abstract
The advancement of the web and information technology has contributed to the rapid growth of digital libraries and automatic machine translation tools which easily translate texts from one language into another. These have increased the content accessible in different languages, which results in easily performing translated plagiarism, which are referred to as “cross-language plagiarism”. Recognition of plagiarism among texts in different languages is more challenging than identifying plagiarism within a corpus written in the same language. This paper proposes a new technique for enhancing English-Arabic cross-language plagiarism detection at the sentence level. This technique is based on semantic and syntactic feature extraction using word order, word embedding and word alignment with multilingual encoders. Those features, and their combination with different machine learning (ML) algorithms, are then used in order to aid the task of classifying sentences as either plagiarized or non-plagiarized. The proposed approach has been deployed and assessed using datasets presented at SemEval-2017. Analysis of experimental data demonstrates that utilizing extracted features and their combinations with various ML classifiers achieves promising results.

1 Introduction
The advancement of the Internet and information technology have expanded rapidly the availability of digital libraries and automatic machine translation tools, which facilitate translating a text from one language to another language. This has caused the number of cases of translated plagiarism, referred to as “cross-language plagiarism”, to perform substantially. It is a type of plagiarism that occurs when textual content is translated into another language without giving acknowledgment of original sources. This type of plagiarism is more difficult to detect since each language has its own structure.

Several plagiarism detection techniques have been proposed to address monolingual plagiarism, that identify plagiarism instances written in the same language. However, there have been few studies that concentrate on researching and developing methods for identifying cross-language (and in particular English-Arabic) plagiarism. These techniques cannot effectively detect more extensively disguised cases of cross-language plagiarism. Eisa et al. (2015) observed that existing techniques have difficulty detecting linguistic modifications like replacing words and phrases by synonyms. When a text is translated from Arabic into English, synonyms are introduced after the translation, thus it is difficult to identify plagiarism.

This paper proposes a new approach for enhancing English-Arabic cross-language plagiarism detection at the sentence level. This technique is based on semantic and syntactic feature extraction using word alignment, word order, word embedding and multilingual encoder models. We investigate the effectiveness of using those features and their combination with different machine learning (ML) algorithms for classifying sentences as either plagiarized or non-plagiarized. The rest of this paper is structured as follows: Section 2 presents a review of related work on cross-language plagiarism detection techniques. In Section 3 we illustrate the proposed approach. The experimental results and discussion are provided in...
Section 4. Finally, Section 5 concludes and presents future work.

2 Related Work

A number of studies have been scrutinized cross-language plagiarism detection. Potthast et al. (2011) presented a classification of cross-language similarity detection methods which was subsequently developed by Danilova (2013). These approaches were classified on the basis of the mechanism used for detecting similarity as shown in Figure 1.

![Cross-language similarity analysis](image)

Figure 1: Classification of various techniques for cross-language similarity analysis (Potthast et al., 2011; Danilova, 2013).

For an instance, Cross-Language Character n-Grams (CL-CNG), which were presented by McNamee and Mayfield (2004), segmenting texts into n-grams for performing comparisons between pairs of texts and measuring the similarity without translation. Another study was on the basis of comparable corpora and was presented by Potthast et al. (2008), who introduced the Cross-Language Explicit Semantic Analysis (CL-ESA) model that were used Wikipedia for computing the similarity between pairs of documents in different languages. For parallel corpora, Barrón-Cedeño et al. (2008) offered Cross-Lingual Alignment-based Similarity model (CL-ASA), creating a bilingual unigram dictionary for comparing pairs of texts. Gupta et al. (2012) introduced the dictionary-based Cross-Language Conceptual Thesaurus Similarity model (CL-CTS) which detects similarity between texts from different languages. Franco-Salvador et al. (2013) introduced a technique based on knowledge graphs for comparing between documents in different languages. Barrón-Cedeño (2013) presented a machine translation model to convert texts into the common language followed by employing a monolingual analysis.

Some published research has focused on English-Arabic cross-language plagiarism detection. For example, Aljohani and Mohd (2014) proposed an English-Arabic cross-language detection approach based on Google Translate to translate the texts and applying a winnowing algorithm, which proposed by Schleimer et al. (2003). Another study presented by Hattab (2015) proposed a technique based on Latent Semantic Indexing (LSI) and parallel corpora to build a cross-language semantic vector space to compute similarity of the context. Alaa et al. (2016) used a logistic regression classifier based on longest common subsequence and cosine similarity measurements and n-gram features at keyphrase level. A study utilized semantic metrics and WordNet for gauging the degree of semantic similarity between words and used it to calculate the similarity for texts and paragraphs of English-French and English-Arabic plagiarism instances Hanane et al. (2016). Ezzikouri et al. (2018) employed a fuzzy semantic approach to identify cross-language plagiarism cases employing Wu and Palmer’s (1994) similarity metrics and WordNet to compute semantic similarity between words.

Based on this review, we have only identified a few studies which have attempted to detect cross-language plagiarism in the English-Arabic domain. Most of these studies have tried to identify plagiarism based on semantic features and key phrases. To the best of our knowledge, none of these studies has tried to detect plagiarism using English-Arabic pairs based on sentence level analysis, nor has any integrated semantic and syntactic features using word embedding and word alignment features with multilingual encoder models.

3 Proposed Method

The key idea of the proposed plagiarism detection technique for English-Arabic pairs of sentence is formulated as a classification task, which classifies each pair of sentences as either plagiarized or non-plagiarized. In order to tackle this problem, it is necessary to analyze texts using different features extracted at syntactic and semantic levels. Thus, we propose methods based on word embedding, word order and word alignment with multilingual
3.1 Feature Extraction

Analysis of semantic and syntactic features forms an essential step for plagiarism detection algorithms. Various sets of extracting features are proposed depending on word embedding, word order and word alignment for pair of sentence comparison. The following subsection describes the extracting features.

3.1.1 Word Order Similarity Features

Syntactic features based on word order are employed in similarity and plagiarism detection algorithms such as those by Li et al. (2006) and Abdi et al. (2015). Therefore, we propose a method that relies on word order features based on machine translation, since word order exhibits beneficial information about the relationships between words. In the case where two sentences have exactly the same words, but in a different order, any approach that measures similarity between texts based on a “bag of words” will show them to be exactly the same. Consequently, the influence of the word order should be taken into consideration when text similarity is computed. Thus, we are motivated by Li et al.’s (2006) approach. However, the proposed method is based on a pre-trained word2vec model released by Mikolov et al. (2013), representing words as vectors that characterize identification of semantic and syntactic features.

In order to gauge word order similarity between pairs of sentences, it is required to convert words into vectors based on a joint word set, which is formed utilizing distinct words from each pair of sentences. For example, given a pair of sentences T1 “A quick brown dog jumps over the lazy fox” and T2 “A quick brown fox jumps over the lazy dog”, a joint word set T contains all distinct. Words from T1 and T2, so T is {A quick brown dog jumps over the lazy fox}. Each word in T1 and T2 has an assigned unique index number, representing the word’s location in the sentence. A word order vector is created for each sentence (r1 and r2 respectively), based on word embedding and the joint word set T. Taking T1 as an example, for each word in T, we look for the same or the most similar word using

1. If the word exists in T1, the value for this word in r1 will take the same index number from T1.
2. If the word does not appear in T1, then we use the pre-trained word2vec model for finding the most similar word using
based on computing cosine similarity between the words. If the similarity score is greater than the predefined threshold (wt), the value of the word in r1 is set to the index number of the word in T1.

3. If the above two processes fail, the value of the index number in r1 is set to 0.

The procedure used for creating r1 will be applied for creating r2, which represents the second sentence. Therefore, word order vectors are constructed as:

\[ r1 = [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9] \]
\[ r2 = [1 \ 2 \ 3 \ 9 \ 5 \ 6 \ 7 \ 8 \ 4] \]

Then, the word order similarity is calculated by using Equation 1.

\[ \text{OrderSim}(r1, r2) = 1 - \frac{\|r1 - r2\|}{\|r1 + r2\|} \] (1)

3.1.2 Sentence Embedding Features

Due to the detection being based at sentence level, extracting features from a pair of sentence uses the technique proposed by Alotaibi and Joy (2020). They proposed an approach for calculating the degree of semantic similarity between two sentences. The authors leverage models that represent sentences embedding, including universal sentence multilingual encoder (MUSE) and averaging word embedding, for constructing sentence vectors. They represent sentence embedding based on: (i) word embedding and term weight schemes (i.e., term frequency inverse document (TFIDF) and part of speech (POS)), referred to as CL-WE-Tw, and (ii) the MUSE model. Based on the methods, computing the degree of semantic similarity between two sentences is by following these steps:

- **Sentence vector based on CL-WE-Tw:** represents vectors for each sentence by taking the average vectors with their weighting according to Equation 2

\[ Sv = \frac{1}{n} \sum_{i=1}^{n} vec \ast (\text{TFIDF} \ast \text{POS}(wi)) \] (2)

where \( Sv \) is sentence embedding, vec is a function that gets word vector, wi is the \( i^{th} \) word of text.

- **Sentence Embedding based on MUSE:** uses a pre-trained model released by Yang et al. (2019) to represent sentence vectors then cosine similarity is employed to measure semantic similarity between pair of vectors as shown in Equation 3.

- **Semantic similarity measure:** after representing vectors for each sentence, cosine similarity is applied to find the degree for pairs of sentence according to Equation 3.

\[ S_{\text{sim}}(veE, veA') = \frac{veE \cdot veA'}{\|veE\| \cdot \|veA'\|} \] (3)

where \( S_{\text{sim}} \) is sentence similarity that calculated using cosine similarity on sentence embedding, \( veE \) is the sentence vector for the English sentence, and \( veA' \) is the sentence vector translated from the Arabic sentence.

- Finally, the authors proposed to integrate semantic similarity features, obtained from the CL-WE-Tw and MUSE methods given by Equation 4.

\[ S_{\text{sentence}} = \frac{(S_{\text{CL-WE-Tw}} + S_{\text{MUSE}})}{2} \] (4)

where \( S_{\text{CL-WE-Tw}} \) is the similarity score obtained from CL-WE-Tw method, and \( S_{\text{MUSE}} \) is obtained from the MUSE model.

3.1.3 Combined Similarity Measures

Since semantic and syntactic features play an important role in interpreting the meaning of a sentence, we propose to combine all sentence similarity measure features, which are described in Section 3.1.1 and CL-WE-Tw, and refer to it as “CL-WET-WO”, as shown in Equation 5.

\[ S(T1, T2) = \delta S_{\text{sim}} + (1 - \delta) \text{OrderSim} \] (5)

Li et al., (2006) suggested that \( 0.5 < \delta \leq 1 \), should be the threshold for weighting significance between components based on word order (OrderSim) and CL-WE_Tw (Ssim).

3.1.4 Word Alignment Features

The word alignment features are employed in different natural language processing tasks like sentence similarity (Sultan et al., 2015) and
paraphrase identification (Mohammad et al., 2017). Therefore, we propose to use semantic based features depending on word alignment. The proposed method is based upon the word alignment approach of Michlase et al. (2006) and Zhou et al. (2019), however, the difference is to use the pre-trained multilingual encoder model, such as that released by Yang et al. (2019) as a bilingual resource. It provides rich semantic information and enables the representation of words from different languages (e.g., English and Arabic) in a single vector space, where it directly determines similarity between words that are written in different languages. Such words are aligned according to their semantic similarity in the model, and cosine similarity is applied to find the similarity between pairs of words. The proposed method consists of two components, that can be used to describe pairs of sentences. The first component finds the similarity score between pairs of sentences as shown in Equation 6, which we call cross-language weighted alignment (CL-WA). This component consists of two processes.

$$S = \frac{1}{2} \left( \frac{\sum_{w \in T_1} \left( \text{maxsim}(w, T_2) \ast \text{idf}(w) \right)}{\sum_{w \in T_1} \text{idf}(w)} + \frac{\sum_{w \in T_2} \left( \text{maxsim}(w, T_1) \ast \text{idf}(w) \right)}{\sum_{w \in T_2} \text{idf}(w)} \right)$$

To compute the semantic similarity between two sentences $T_1$ and $T_2$, we use the pre-trained multilingual encoder model instead of a monolingual dictionary, then cosine similarity is employed for measuring similarity between pairs of words. The following steps are used.

1. According to Equation 7, for each term in sentence $T_1$ we determine its aligned word in the sentence $T_2$ which gets the highest semantic similarity and is greater than the threshold ($t_1$). This threshold is suggested to avoid excessive noise that leads to deterioration of overall performance. For example, when we align word “يضع” which means ‘put’ in English language, with other words like “dance”, “put” and “cook”, we find their vectors such as $[(\text{vector (dance)}), (\text{vector (put)}), (\text{vector (cooking)})], (\text{vector (cook)}), (\text{vector (يضع)})]$, then we determine the maximum degree of semantic similarity between their vectors by applying cosine similarity.

$$\text{maxsim}(w, T_2) = \text{maxsim}(w, T_2)$$

2. Determine the importance of words in $T_1$ using inverse document frequency (idf).

The same process is employed to determine the most similar word in $T_1$ beginning with words in $T_2$. Finally, the similarity between the input sentence $T_1$ and $T_2$ is computed using Equation 6.

The second component is to calculate the semantic similarity for given two sentences $T_1$ and $T_2$ according to Equation 8, which we call “cross-language alignment (CL-A)”. The process is as follows.

1- For each term in an Arabic sentence, we try to determine the word in the English sentence that has the highest semantic similarity (i.e., using the pre-trained multilingual encoder model and employing cosine similarity) that is greater than the threshold ($t_2$). This threshold is suggested to avoid excessive noise which causes deterioration of overall performance.

2- Finally, we take the average score over all the maximal similarity scores as given by Equation 8.

$$ssim(S, T) = \frac{1}{m} \sum_{i=1}^{m} \text{max Cos}(si, tj)$$

Finally, the overall sentence similarity score is computed based on CL-WA and CL-A, which we name “CL-WA+CL-A”, as shown in Equation 9.

$$\text{Sentence}_{sim} = th \ast S + (1 - th)ssim$$

where the value of $th$ value ranges within $[0.5, 1]$, and is the threshold for weighting importance between components based on CL-WA ($S$) and CL-A ($ssim$).

### 3.2 Classification Model

Machine learning algorithms have been applied in several fields, such as image processing and natural language processing. We use the extracted features, based on syntactic and semantic computation, along with different ML classification frameworks for detecting whether an English-Arabic pair of sentences is plagiarized or not. We investigate
different ML classifiers such as Logistic Regression (LR), Support Vector Classifier (SVC), Linear Support Vector Classifier (LSVC), Decision Tree (DT), Random Forest (RF), K Nearest Neighbors (KNN), and Extreme Gradient Boosting (XGBoost), using those extracted features.

4 Experiments and Results

To assess the performance of the proposed techniques, we have conducted experiments to examine the impact of both individual and combined features used to train each classifier. For evaluating the performance of the classifiers, we have used 10-fold cross-validation and the F1-measure (F1 score), which is the harmonic average of precision and recall, as shown in Equation 10. The experiments have been carried out using Python with the scikit-learn library to build each classifier, and we have used the Grid Search method to find the best values of hyperparameters for configuration of the ML models.

\[
F - measurement = \frac{2 \times precision \times recall}{precision + recall}
\]  

(10)

4.1 Dataset

We used SemEval-2017 (Cross-lingual Arabic-English) datasets, released by Cer et al. (2017). The total size of the dataset is 1234 pairs of sentences, which were used for both training and testing data. Humans have labeled each pair of sentences on an integer scale from 0-5 (5 indicates exactly similar, whereas 0 shows that the two sentences in the pair are completely different), which was linearly scaled into the interval [0,1] then each pair of sentences was labeled 1 (means plagiarized) or 0 (means non-plagiarized) if the human similarity score is greater than or equal to a threshold of 0.5. Table 1 illustrates more information about the dataset.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Source</th>
<th>Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSRvid</td>
<td>MSR-Video, Microsoft Research Video Description Corpus</td>
<td>735</td>
</tr>
<tr>
<td>SNLI</td>
<td>Stanford Natural Language Inference corpus</td>
<td>250</td>
</tr>
<tr>
<td>SMTeu</td>
<td>WMT2008 development</td>
<td>149</td>
</tr>
<tr>
<td>MSR-Para</td>
<td>Microsoft Research Paraphrase Corpus</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 1: Evaluation dataset

4.2 Pre-processing Stage

The pre-processing phase is an essential step for preparing the text for further evaluation. As the first two extracted features described in 3.1.1 and CL-WE_Tw are based on machine translation, we used the Google Translation tool to translate Arabic sentences into English sentences. Then, we used the Natural Language Toolkit (NLTK) tool for the following processing: (1) tokenization, (2) part of speech tagging, (3) removing punctuation marks and (4) normalization. On the other hand, the extracted features described in 3.1.4 are based on a pre-trained multilingual encoder model, which represents different languages on the same vector space, therefore, we used NLTK for performing: (1) tokenization, (2) removing punctuation marks and (3) removing stop words for both English (e.g. ‘that’, ‘is’ and ‘were’) and Arabic such as ‘في’، ‘ الى’ and ‘التي’، meaning “in”, “to” and “that” in English respectively.

4.3 Parameters Setting

The proposed methods of extracting features contain a number of parameters that are required to be tuned parameters for constructing word order vectors, for weighting the importance between syntactic and semantic features, and for word alignment features. Many experiments are performed to determine a suitable value for each parameter. For setting these parameters, we have used pairs of sentences from the Microsoft Research Video Description Corpus dataset, and computed Pearson correlation coefficient between human rates and results obtained from proposed approaches, thus the best Pearson correlation indicates suitable values for these parameters. Therefore, the results acquired from the experiments show the suitable values of the parameters, and we have found the best correlation coefficient values for determining word order similarity is achieved at (wt= 0.54). For weighting importance of syntactic and semantic similarity features, we have obtained the best results at (δ=0.80). In terms of the parameters related to word alignment, the best performance is attained at (t₁= 0.53, t₂= 0.40 and th=0.70). As a result, we
have used these parameters and values on the rest of the dataset.

4.4 Results

This section shows the contribution of using extracted features, described in Section 3.1, along with a set of classifiers for detecting cross-language plagiarism. Table 2 shows the performance results of utilizing the proposed features along with LR, SVC, L SVC, DT, RF, KNN and XGBoost classifiers, where the first column illustrates the extracted features while the rest of the columns presents the results according to the F1 Score metric.

4.5 Discussion

As presented in Table 2, the performance of the classifiers using sets of extracted features shows encouraging results for classifying pairs of English-Arabic sentences. We can see that the integration of semantic and syntactic features with the classifiers as one feature, based on word embedding and word order features, demonstrates enhancement of the performance through LR, L SVC, KNN and XGBoost. Furthermore, it can be observed that using CL-WA+CL-A features along with the different classifiers obtained better results than CL-WA and CL-A individually. It can be also seen that combinations of features based on CL-WET-WO and CL-WA+CL-A with the classifiers show improvements in the results. Interestingly, using all combined features based on CL-WET-WO, CL-WA+CL-A and CL-WE-Tw+MUSE is efficient in enhancing the performance of most classifiers including LR, SVC, L SVC, DT and RF. We believe this improvement can be ascribed to the word embedding and multilingual encoder models capturing semantic and syntactic features.

5 Conclusion

In this paper we introduced a technique based on analyzing sentences using syntactic and semantic features with ML classifiers to detect English-Arabic cross-lingual plagiarism. The features we used involve word order, word embedding and word alignment with multilingual encoders. We also explored the effects of using extracted features and their combinations along with the different classifiers. The proposed method has been assessed by using a compilation of four datasets. According to the evaluation, the integration of combined extracted features with the classifiers demonstrates improved performance. Overall, the SVC classifier based on combination of all features accomplishes the best results with the F1 score of 0.879. In future work, the approach will be expanded to include use of neural network techniques.

Table 2: Classification results based on set of extracted features.

<table>
<thead>
<tr>
<th>Features</th>
<th>Classifier</th>
<th>LR</th>
<th>SVC</th>
<th>L SVC</th>
<th>DT</th>
<th>RF</th>
<th>KNN</th>
<th>XGBoost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1- Word order</td>
<td></td>
<td>0.786</td>
<td>0.776</td>
<td>0.783</td>
<td>0.716</td>
<td>0.728</td>
<td>0.756</td>
<td>0.764</td>
</tr>
<tr>
<td>2- Word Embedding</td>
<td></td>
<td>0.839</td>
<td>0.845</td>
<td>0.839</td>
<td>0.770</td>
<td>0.770</td>
<td>0.824</td>
<td>0.834</td>
</tr>
<tr>
<td>3- CL-WET-WO</td>
<td></td>
<td>0.847</td>
<td>0.842</td>
<td>0.846</td>
<td>0.766</td>
<td>0.766</td>
<td>0.833</td>
<td>0.845</td>
</tr>
<tr>
<td>4- CL-WE-Tw+MUSE</td>
<td></td>
<td>0.8561</td>
<td>0.859</td>
<td>0.857</td>
<td>0.812</td>
<td>0.813</td>
<td>0.851</td>
<td>0.865</td>
</tr>
<tr>
<td>5- CL-WA</td>
<td></td>
<td>0.812</td>
<td>0.815</td>
<td>0.810</td>
<td>0.743</td>
<td>0.750</td>
<td>0.784</td>
<td>0.817</td>
</tr>
<tr>
<td>6- CL-A</td>
<td></td>
<td>0.768</td>
<td>0.772</td>
<td>0.766</td>
<td>0.689</td>
<td>0.711</td>
<td>0.753</td>
<td>0.788</td>
</tr>
<tr>
<td>7- CL-WA + CL-A</td>
<td></td>
<td>0.818</td>
<td>0.822</td>
<td>0.826</td>
<td>0.743</td>
<td>0.755</td>
<td>0.804</td>
<td>0.821</td>
</tr>
<tr>
<td>8- Features (3 and 7)</td>
<td></td>
<td>0.853</td>
<td>0.850</td>
<td>0.857</td>
<td>0.789</td>
<td>0.814</td>
<td>0.846</td>
<td>0.844</td>
</tr>
<tr>
<td>9- Features (3, 4 and 7)</td>
<td></td>
<td>0.871</td>
<td><strong>0.879</strong></td>
<td>0.875</td>
<td>0.853</td>
<td>0.861</td>
<td>0.852</td>
<td>0.864</td>
</tr>
</tbody>
</table>

References


Sultan, M.A., Bethard, S. & Sumner, T. (2015). Dls@cu: Sentence similarity from word alignment and


Towards a Better Understanding of Noise in Natural Language Processing

Khetam Al Sharou∗, Zhenhao Li∗, Lucia Specia
Language and Multimodal AI (LAMA) Lab, Imperial College London, UK
{k.al-sharou, zhenhao.li18, l.specia}@imperial.ac.uk

Abstract

In this paper, we propose a definition and taxonomy of various types of non-standard textual content – generally referred to as “noise” – in Natural Language Processing (NLP). While data pre-processing is undoubtedly important in NLP, especially when dealing with user-generated content, a broader understanding of different sources of noise and how to deal with them is an aspect that has been largely neglected. We provide a comprehensive list of potential sources of noise, categorise and describe them, and show the impact of a subset of standard pre-processing strategies on different tasks. Our main goal is to raise awareness of non-standard content – which should not always be considered as “noise” – and of the need for careful, task-dependent pre-processing. This is an alternative to blanket, all-encompassing solutions generally applied by researchers through “standard” pre-processing pipelines. The intention is for this categorisation to serve as a point of reference to support NLP researchers in devising strategies to clean, normalise or embrace non-standard content.

1 Introduction

In Natural Language Processing (NLP), “noise” as a concept is not well understood and often described as hard to define (Taghipour et al., 2011). In this paper, we attempt to outline what different types of noise mean in order to support NLP researchers in devising strategies to clean, normalise or embrace unexpected content at either training or inference time.

The term “noise” has been generally used to cover both harmful and meaningful types of non-standard content. We however believe that a distinction should be made between content that should be removed/normalised because it harms NLP systems – henceforth harmful noise – and content that should be kept to improve the performance of such systems – henceforth useful noise1. To clarify further, the usual approach to handling “noise” is to attempt to clean or normalise the data as much as possible, but this content can be as important as any other elements in the data because it carries meaning or intentions. In text classification applications, for example, some types of noise should be kept and taken into account, e.g. certain punctuation patterns in sentiment analysis. Furthermore, in generation applications, noise often needs to be transferred to the output, such as emojis in machine translation to preserve sentiment. Moreover, in tasks such as correction of second language learners’ essays, errors such as lack of cohesion and incorrect punctuation need to be retained.

Handling noise in NLP has been attracting significant attention especially with the widespread availability of data from social media platforms such as Twitter, Reddit, among others, that created not only new opportunities but also new and different needs (Karpukhin et al., 2019). Text found on these platforms, as well as other user-generated materials, is full of non-standard content (Eisenstein, 2013) which causes problems to NLP systems – typically trained on clean data – and these fail to handle them correctly (Baldwin et al., 2015; Belinkov and Bisk, 2018; Heigold et al., 2018). Dealing with non-standard data has been targeted as a research direction in some areas, for example, machine translation (MT), with methods that aim to be robust to unexpected noise (Belinkov and Bisk, 2018; Karpukhin et al., 2019; Vaibhav et al., 2019). However, work has been mainly limited to a few sources of artificially injected noise types.

1We will keep using the term “noise” to adhere to a term known and used in the existing literature in NLP, but we make a distinction between harmful noise and useful noise.
In this paper, we start by providing an overview of previous work addressing this issue (Section 2); to then propose a definition of noise including cases that needs to be cleaned/normalised (harmful noise) and other cases that needs to be kept (useful noise), and present a comprehensive taxonomy of types of noise (Section 3); and to finally demonstrate the impact of addressing noise in a task-based fashion through experimenting with three tasks, where we compare filtering out harmful noise and keeping useful noise versus cleaning/normalising or keeping all types of noise (Section 4).

Our focus is on linguistic noise rather than other types of noise (such as numeric mislabelling). Therefore, our definitions and categorisation broadly cover (i) text classification/regression tasks, where the noise is in the input text, e.g. sentiment analysis; and (ii) text-to-text tasks, where the noise is in the input text and – at training time – can also be in the output text, e.g. machine translation.

2 Related Work

Noise in NLP has typically been defined in context of a specific dataset, intermediate or end-task, rather than in general. For example, noise is limited to sentence pairs “not being parallel” in parallel corpus filtering (Taghipour et al., 2011). Han and Baldwin (2011) describe their task as lexical normalisation of “ill-formed words”, where the definition is narrowed down to “instances of typos, ad hoc abbreviations, unconventional spellings, phonetic substitutions and other causes of lexical deviation found on Twitter and SMS messages”. Subramaniam et al. (2009) define text normalisation where noise becomes “any kind of difference in the surface form of an electronic text from the intended, correct or original text”. For an end task where the focus is on improving the performance of a certain task, however, the definition of noise becomes very specific, serving one purpose. For example, noise is referred to as the task of removing unconventional casing by truecasing the data when the aim is to get better vocabulary generalisation. Very often such a blanket, one-size-fits-all pre-processing pipeline is applied without enough consideration to other possible sources of noise, or which sources should actually be kept in the data.

Our work is driven by the lack of studies aimed at defining and providing a comprehensive categorisation of noise types in a way that reflects a better practice and shows sensitivity towards such content. The few exceptions limit their categories to noise found in specific text types. Subramaniam et al. (2009) provide a general overview of noise types in SMS, emails and online chats, while Eisenstein (2013) focuses on identifying types of what is called bad language and their possible causes.

Most of the previous studies on handling noise have focused on MT. While some studies have revealed that training with noise increases the robustness of systems towards noisy data, others have indicated that the quality of their systems degraded. This can be attributed to the differences in noise types and their potential impact on the task. For instance, Khayrallah and Koehn (2018) have showed that training MT models on noisy parallel data (such as misaligned sentences or wrong language) leads to weaker systems. However, when models are trained on more specific types of noise such as spelling, profanity, grammar and emoticons, they have been shown to perform better on similar types of noise (Belinkov and Bisk, 2018; Heigold et al., 2018; Ott et al., 2018; Berard et al., 2019; Vaibhav et al., 2019). We note that most of the latter work aims to adapt MT systems built on clean data to noisy test settings by artificially injecting errors to the training set. This is a different problem to the one we discuss in this paper where both training and test sets are taken from the same larger population with naturally occurring noise.

These studies show that different types of noise have different effects in MT and other NLP tasks. However, the types of noise addressed and their effect vary considerably and a consistent definition has not been provided.

Noise is better defined and understood in other related fields. In Automatic Speech Recognition, a distinction is made between what is considered as noise (e.g. environmental noise, reverberation) and other speech variations (e.g. accent, speaking style and rate, emotional states, speech impairments). Noise is defined as “any unwanted disturbances superposed upon the intended speech signal” (Li et al., 2015), e.g. by additive noise (e.g. background noise, traffic noise) and convolutional noise (e.g. transmission channel distortions, room reverberation, microphone filtering) (Xiong, 2009). In Optical Character Recognition, noise refers to “the error in the pixel value or an unwanted bit pattern, which do not have any significance in the output”, where the unwanted bit patterns are introduced by uneven writing surface or poor quality of the data.
acquisition device and include textured or coloured background (Bansal and Kumar, 2013). Different handwriting however is not seen as noise which needs removing but as a problem which needs to be addressed.

Similar to what has been done in these areas, this paper is an attempt to clearly define and categorise noise in NLP in a way that is not constrained to any particular application, task or dataset. While noise is mainly found in non-standard text, the taxonomy is agnostic to the type of text.

3 A Taxonomy of Noise

We intend to move away from the general labelling of noise as “errors”, “difficult data”, “ill-formed words”, “contamination”, “disfluent language” and “corruption” to a more specific one that reflects the current practice which have shifted from denoising to training with noisy text. We therefore look at noise through the same lens as in other related fields where a distinction should be made between unwanted constructs that occur in the text (generally unintentionally), and other (generally intentional) constructs that do not adhere to the conventions and rules of a given language to serve a purpose. Based on this distinction, we define harmful noise as any unwanted disturbances that are either harmful or useless or both: Harmful in that it affects the intended meaning of the text and/or the performance of the NLP system; and useless in that their existence does not serve a purpose. Useful noise, on the contrary, covers wanted content that serves a purpose and carries meaning that is important for the task and/or for the performance of the NLP system. Our definition does not includes cases where a special language is used in a cryptical fashion, i.e. hiding some codes/messages in the text so that only a particular addressee understands the true message. It is also worth noting that it depends on the NLP application under consideration, where harmful noise for one system could be useful noise for another one.

Our taxonomy of noise, displayed in Figure 1, focuses on applications that take sentences as input, as in most NLP tasks. From this work, we exclude types of noise that are related to errors in parallel corpora with text in both input and output such as misalignments, incorrect output, untranslated sentences, among others. The taxonomy is divided into several types and sub-types of noise that are limited to those naturally occurring, produced by humans. Machine-generated errors, e.g. stemming from back-translation or intentionally injected, are seen as a way to mimic the naturally occurring noise, and as such not represented separately. The taxonomy is designed to serve as an initial point of reference. It should be seen as language-independent, but it may require modifications to cover very specific language phenomena the authors are not familiar with. Our taxonomy is based on previous works that address different types of noise as well as taxonomies of errors and also on our wide experience improving robustness in NLP. The selection criteria are driven by the intention to include general categories that are applicable to different languages and tasks, and for non-standard content. In what follows, we describe the types and sub-types of noise coming under the proposed taxonomy and provide examples.

Orthography: This type of noise is concerned with the way words are written. Several sub-types come under this type, which we describe below. Some of them are considered as errors, e.g. spelling errors, while others are looked at as variations or deviations from the standard way of writing to serve a purpose, e.g. word obfuscation, word lengthening.

Spelling Errors: This is when a word is spelt in a way that deviates from reference dictionaries, standardised or accepted norms, or recognised usage. The misspelling of a word takes different forms. For example, the word “receive” can be misspelt by deleting a character, e.g. “receve”; inserting an extra character, e.g. “recieve”; swapping adjacent characters, e.g. “recieve”; or replacing a character with another, e.g. “recieve” (Sakaguchi et al., 2017; Belinkov and Bisk, 2018). Additionally, a spelling error can occur when writing a word without a hyphen where needed or with a space where it should be written as one word such as writing “4 MB” instead of “4MB” (Bušta et al., 2009).

Orthographic Variants: This covers the spelling of words in different ways due to: 1) regional variations: e.g. British English spelling vs. American English spelling, e.g. “centre”, “center”; 2) words with different correct spellings: e.g. “spelled”, “spelt”, “سورية” or “سورية” for “Syria” in Arabic, or when words are transliterated, e.g. proper name “محمد”, having one spelling in Arabic, could be written as “Muhamed”, “Mohamed” or “Mohammed” in English; or 3) diacritical marks:
in some languages, such as Arabic, words have accents (or what is called diacritics), which are not always used. These diacritics can change the meaning of the word. For example, the word “جد” could mean “grandfather” with the diacritical mark placed above the first letter “جد jed” and “dil- lIGENCE” if this mark is placed under the first letter “جد jid”. It is therefore important that the models are trained to recognise the difference.

**Casing:** This sub-type refers to instances where casing is used for a purpose. Some words are capitalised for emphasis, e.g. “NOTED!” It also covers cases where casing is incorrect or missing where needed, generated by mistake and not deliberately to serve a purpose such as random capitalisation of some characters in a word, e.g. “SUre”; or absence of casing where needed, for example, on proper names, e.g. “john”.

**Word Obfuscation:** This sub-type refers to cases where some characters within a word are obfuscated, or in other words, disguised, using numbers or symbols. It can be used for purposes such as disguising violence, e.g. “ki11” instead of “kill”; or masking profanity, e.g. writing “fuck” as “f*ck” (Michel and Neubig, 2018).

**Word Lengthening:** It refers to elongating a word by replicating a letter(s) in it, often to express emphasis, e.g. “Yes, Nooow!”; or sentiment, e.g. “قمرور” which means “moon” in Arabic and used to compliment a girl’s beauty.

**Symbols:** This covers any special symbol or sign used to express an idea, mood or feelings, e.g. emoticons “:-)”, emojis “ energía”; or as a replacement for a word, e.g. using “@” to mean “at”.

**Informal Word Forms:** This type refers to cases where multiple words are written jointly as one word, following a certain dialectal convention (Subramaniam et al., 2009), for example, dialectal words or slang, e.g. “wanna” for “want to”, “whatcha” for “What are/do you ...?”. This form of contraction does not normally follow the conventions of how words are contracted and is different from other more common forms of contraction such as “isn’t” for “is not” and “aren’t” for “are not”. In the extreme case, most of the text could be written in dialect such as in Arabic (Darwish, 2014).

**Shortening:** This type refers to cases where a word or phrase is written in a short form using different techniques, including three sub-types: 

- **Abbreviations** includes any form of shortening of a word or phrase used to refer to the whole word or phrase, for example, “Professor” is abbreviated as “Prof”. It also covers acronyms such as Internet slang, e.g. “LOL” for “Laugh Out Loud”; and initials, e.g. “idk” for “I don’t know” (Subramaniam et al., 2009).

- **Deletion** refers to cases where words and phrases are shortened without following any well-established patterns (i.e. more arbitrarily), for example, by character deletion, e.g. “msg” for “message”; by cutting part of a word, i.e. truncation, e.g. “tom” for “tomorrow”; or deleting an entire
word, e.g. “drvng hm” for “I am driving home” (Subramaniam et al., 2009).

**Substitution** happens when words or characters are replaced with numbers or letters which have the same phonetic sound to make it shorter. Substitutions may encompass several sounds. Examples include writing “2day” for “today”, “18r” for “later”, and “byk” for “bike” (Subramaniam et al., 2009; Gouws et al., 2011; Han and Baldwin, 2011). The use of numerals in place of letters can also happen for other proposes, e.g. writing Arabic text in Latin letters and using Arabic numerals to represent letters when there is no equivalent in the Latin script (Darwish, 2014). For example, the word “تحرير” which means “liberty” could be written as “ta7r¯ır” with the letter “h” being replaced with number “7” (ibid.). Another type of substitution errors occurring in texts includes when typing wrong keys on a keyboard instead of the intended ones (Kane et al., 2008) (this could also be seen as a spelling error, see Orthography type).

In some instances, shortening can using a mix of techniques, for example, by both deletion and substitution, e.g. “f2f” for “face-to-face”.

**Grammatical Errors:** This type of noise implies a deviation from the grammatical rules of a language apart from spelling errors (Lommel and Melby, 2015; Garnier and Saint-Dizier, 2016), including the following sub-types:

**Function Words:** Function words include prepositions, articles, determiners that are used incorrectly (Lommel and Melby, 2015), e.g. wrong preposition, “I bought this book to her” instead of “I bought this book for her”.

**Word Form:** This sub-type refers to a problem in the form of a word and includes agreement, tense-mood-aspect, and part-of-speech (Lommel and Melby, 2015), e.g. “I have a good day yesterday” (present tense) instead of “I had a good day yesterday” (past tense).

**Word Order:** This sub-type refers to instances where the order of words is incorrect (Lommel and Melby, 2015). For example, unlike in English, in Arabic, an adjective comes after a noun to describe it, so it is incorrect to say “a big house” where it must be “a house big”.

**Incorrect Punctuation:** Punctuation errors may include missing or incorrect placement of punctuation marks (e.g. !, ?, etc.) (Bušta et al., 2009). Punctuation plays a major role in our understanding of a text and text readability. For example, the sentence “Eat, dog!” could be read and interpreted differently with or without a comma.

**Cohesion Errors:** This type generally refers to structural errors that affect the flow of the text (within or across sentences) caused by using wrong linking words or pronouns, e.g. “Your car is newer, hence mine is faster”.

**Disfluencies in Human (transcribed) Data:** This type covers disfluencies that occur in spontaneous spoken language (Shriberg, 1994), including:

**Pause-filling Words:** This type refers to words or phrases used to express pausing in writing that mimics natural speech, e.g. “uh”, “er”, “um”. They generally do not have any meaning on themselves but may be indicative of important aspects, e.g. hesitation or surprise reaction (positive or negative), as well as style.

**Repetition of Words:** It refers to the occurrence of the same words several times or syntactically similar units unintentionally or on purpose, e.g. “I have I have discussed this matter matter with her again. Still, she is not convinced”.

**Repetition of Punctuation:** Unlike incorrect punctuation sub-type, this type refers to instances where punctuation marks such as exclamation mark or question mark are repeated to serve a purpose, i.e. for emphasis, e.g. “Really! You want me to go now???”; or to express an emotional state, e.g. “What???? This is really annoying!!!”.

**Code-switching:** This type refers to the alternation between different languages in a single sample. For example, “努力” is a phrase mixed with Chinese and English texts, which means “working hard”, with the English suffix “-ing” added to the Chinese word. An entire word or phrase could also be in a different language.

**Internet Jargon:** This type refers to new words and acronyms that gained special meaning and usage in certain social media platforms. Words such as “downvote”, “upvote”, and acronyms such as “TIL” for “Today I Learned”, “OP” for “Original Poster” are examples of jargon found on Reddit (Berard et al., 2019).
**4 Experimental Setup**

This section describes the settings of the experiments we carried out so as to show the impact of different pre-processing strategies on different NLP tasks. We experiment on three tasks based on data (in English) collected from Twitter: Offensive Language Identification, Informative COVID-19 Tweets Identification, and Tweets Sentiment Analysis tasks. We start by describing each task and the data used (Section 4.1), followed by the pre-processing steps of different sources of noise (Section 4.2), and the model architectures (Section 4.3).

Due to length restrictions, we limit ourselves to experimenting with a few types of noise, with the aim to show their role as of being “harmful noise” to filter out or “useful noise” to keep, and to what extent this is dependent on the task.

**4.1 Tasks and Datasets**

Since it is not possible to find corpora that cover all types of noise as defined in the taxonomy, we select three datasets sourced from social media, where the texts are informal and contain several common types of noise. We use the following tasks and their respective freely available datasets (statistics in Table 1):

**Offensive Language Identification (OLID)** (Zampieri et al., 2019a,b): This task focuses on the problem of identifying and categorising offensive language on Social Media. We take the main type of annotation and treat the task as a binary classification task. Given the lack of standard training/development splits in the OLID dataset, we randomly split the training data into training and development sets with a ratio of 0.8/0.2. The official test set in this dataset is used for evaluation.

**Informative COVID-19 Tweets Identification (COVID)** (Nguyen et al., 2020): This is a binary classification task identifying whether a COVID-19 related Tweet is informative or not. We use the official train/dev/test splits in the dataset.

**Twitter US Airline Sentiment Analysis (SA):**

This consists of annotated user reviews on Twitter classified into positive, negative and neutral. We filter the data by only including the annotations where the sentiment confidence is 1, and then randomly split the data into train/dev/test sets with a ratio of 0.8/0.1/0.1.

**4.2 Pre-processing**

The pre-processing step is where we generally make decisions on how to deal with the data in terms of whether to clean, normalise, or keep the data as it is. We experiment with different pre-processing strategies for each task by removing, normalising or keeping seven types and sub-types of noise listed in our taxonomy: casing, @mention tag, hashtag, emoji, code-switching, URL and punctuation. For casing, the pre-processing involves normalising all characters to lowercase. For hashtags and emojis, pre-processing can either remove or normalise them by transforming them into corresponding word phrases that share the same semantic meaning (e.g. “#PutUpOrShutUp” transformed into “Put Up Or Shut Up”, the emoji “😉” transformed into “smiling face”). For the other types of noise, pre-processing means removing them. In this work, we apply the same pre-processing steps to the training, development and test sets.

**4.3 Model and Hyperparameters**

We used a pre-trained BERT (Devlin et al., 2019) model with the “bert-base-cased” architecture\(^3\) as our classifier. A dropout with a rate of 0.1 is applied to the output layer on top of the pre-trained model. Models were trained for 5 epochs on the training set, and the checkpoint with highest macro F1 score on the development set was selected for evaluation on the test set. We fine-tuned BERT using AdamW (Loshchilov and Hutter, 2019) optimiser with a learning rate of $1e^{-5}$. All models were trained on a single V100 GPU, with a batch size of 128 sentences. Our code was based on the [https://www.kaggle.com/crowdflower/twitter-airline-sentiment](https://www.kaggle.com/crowdflower/twitter-airline-sentiment)

\(^3\)We did not use the uncased BERT because it treats the uncased text the same way as its cased version, which would not allow for the comparisons by normalising/keeping the casing information.
Table 1: Number of sentences in the three datasets. For the sentiment analysis data, we report the statistics after filtering. OFF: label “offensive”. NOT: label “non-offensive”. INFOR: label “informative”. UNINF: label “uninformative”. POS: label “positive”. NEG: label “negative”. NEU: label “neutral”.

<table>
<thead>
<tr>
<th></th>
<th>OLID</th>
<th>COVID</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OFF</td>
<td>NOT</td>
<td>UNFIN</td>
</tr>
<tr>
<td>Train</td>
<td>3,518</td>
<td>7,074</td>
<td>3,273</td>
</tr>
<tr>
<td>Dev</td>
<td>882</td>
<td>1,766</td>
<td>472</td>
</tr>
<tr>
<td>Test</td>
<td>240</td>
<td>620</td>
<td>944</td>
</tr>
</tbody>
</table>

Table 2: Baseline results in macro-F1 scores on the three tasks.

<table>
<thead>
<tr>
<th></th>
<th>OLID</th>
<th>COVID</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>baseline</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.793 ± 0.009</td>
<td>0.867 ± 0.002</td>
<td>0.853 ± 0.026</td>
</tr>
</tbody>
</table>

Figure 2: Percentage change in macro-F1 score with the baseline (no pre-processing) when removing one type of noise at a time.

5 Results

We first present our baseline performance without any pre-processing in Table 2. We then show the percentage change in macro-F1 scores for each of the three tasks by addressing one type of noise from the baseline at a time in Figure 2. Each time one type of noise is removed in the pre-processing except the casing, which is lowercased. We consider noise to be “useful” noise when the removal results in decrease in performance (negative bars). It is worth noting that if the performance drops, that means removing the noise might deviate the intended meaning of the original sentence.

Comparing the three tasks, removing/normalising the same type of noise clearly has opposite effects (e.g., casing), or different magnitudes of effect (e.g., punctuation):

Normalising casing leads to a decrease on OLID task whereas the performance on the other two tasks increases. This is intuitive as offensive texts are likely to be written in uppercase while it might be less useful for identifying informative tweets or sentiment (i.e., for sentiment, when both strong positive and strong negative texts involve upercasing, the casing information does not indicate sentiment correctly). A similar trend can be witnessed when removing URLs, but the decrease on OLID task is larger. This might be because, in the OLID dataset, web addresses have already been normalised with a unified “URL” token, thus it might contain useful rather than noisy information. We can therefore say that casing and URLs are useful noise and should be kept on OLID task for better performance.

Hashtags are more useful on the sentiment analysis task because, when removed, the performance drops on the SA task while improves on the other two tasks. We notice that in the data for sentiment analysis, hashtags are mostly single words such as “#mad” and “#senseless” so that the sentiment could be detected by the model. However, in the OLID dataset, hashtags hinting toxicity mostly include multiple words, e.g., “#Liberalismisamentaldisorder”, which increases the difficulty of utilising these hashtags. In the COVID dataset, almost all tweets consist of hashtags related to COVID-19, thus the hashtags do not help identifying whether the tweet is informative or not.

Removing @mentions causes a more obvious decrease in the SA task than the other two tasks. This is because the @mentions in OLID and COVID datasets have been normalised with a unified “@USER” token, but in the SA data @mentions stay in the form of usernames. We found that the @mentions in sentiment analysis data helps indicate the sentiment. For example, 32.0% of sentences with “@VirginAmerica” are labeled as...
positive while there are only 10.9% positive tweets with “@united”. Similarly, removing emojis, code-switching and punctuation leads to a decrease on all three tasks. However, emojis are less influential for identifying informative tweets. The decrease therefore is not significant for this specific task. Furthermore, as the non-English words are mostly named entities, the removal of code-switching could break the sentence structure. Regarding punctuation, it is important for the three tasks where its removal causes larger performance drop especially on the SA task as it leads to the removal of emoticons “:-)”, which can be useful for classifying sentiment.

Based on our findings that show how different strategies lead to different results, we took a step further to show the validity of our reasoning for how noise should be understood and handled on different NLP tasks. To that end, we combined the different pre-processing strategies and trained two other systems: remove all, which does the lower-casing and removes all other types of “noise” we dealt with in our experiment (i.e. URLs, hashtags, @mentions, emojis, code-switching and punctuation), and remove+keep, which only removes the harmful noise in the specific task as showed in Figure 2 for each type of “noise” (e.g. for OLID task only hashtag is removed while for SA task, URL is removed and the texts are lowercased). In addition, to make use of the potentially useful information in hashtags and emojis, we followed the state-of-the-art approaches (Liu et al., 2019; Kumar and Singh, 2020) and segmented hashtags into separate words and transformed emojis into corresponding English phrases as we stated in Section 4.2 (pre-processing of other types of noise is the same as remove+keep). The system trained on this data is noted as remove+keep+transform. The results are presented in Figure 3.

The system with data removing all sources of noise in the pre-processing shows a poorer performance than the baseline, which keeps all sources of noise. However, both “remove+keep” and “remove+keep+transform” systems outperform the baseline, with improvement on sentiment analysis task being the most significant. However, after transforming hashtags and emojis into English phrases, the performance only improves on the OLID task compared to the “remove+keep” system, which confirms our claim that the same noise normalisation strategy to noise might have different influence on different tasks.

These results are in line with our proposed definition of noise in NLP where some types of noise can be useful and others harmful, and how this is greatly dependent on the task. They also confirm our suggestion that one should not simply follow “standard” pre-processing pipelines, but carefully devise appropriate strategies to deal with different types of noise depending on the task.

6 Conclusions

In this paper, we proposed a definition and taxonomy of noise in NLP so as to serve as a point of reference for NLP researchers to consult when they devise strategies to clean, normalise, or embrace non-standard content at either training or inference time to improve the robustness of their systems to this unseen or unexpected naturally occurring harmful and useful noise. We highlighted that noise in NLP should be carefully handled in light of what we call “harmful noise” that needs to be removed when it affects the performance of the system and/or it does not carry the intended meaning of the text, and “useful noise” that needs to be kept because it is an integral part of the data and useful for a task, or even should be added to the training data when it only happens naturally at test time.

Our experiments support our argument by demonstrating that tailored approaches are better than blanket, all-encompassing solutions generally applied by researchers through “standard” pre-processing pipelines. For instance, we found out
that casing and URLs are useful noise and should be kept on OLID task, but having a negative impact on the other two tasks (SA and COVID tasks) and should therefore be removed. We have also shown how special handling of harmful and useful noise could result in better performance where remove-all and keep-all approaches resulted in poorer performance. Our approach to noise was based on their impact on the tasks - we removed types of noise which had negative influence on the tasks. Our goals were to bring awareness to the different types of unexpected content and provide a definition and a taxonomy, and to highlight the fact that they need to be handled carefully rather than being avoided or treated using the same strategies.

References


Comparing Supervised Machine Learning Techniques for Genre Analysis in Software Engineering Research Articles

Felipe A. de Britto¹, Thiago C. Ferreira², Leonardo P. Nunes², and Fernando S. Parreiras¹
FUMEC University¹
Federal University of Minas Gerais²
felipebritto@gmail.com, thiagocf05@ufmg.br, leopereirananunes@gmail.com, fernando.parreiras@fumec.br

Abstract

Written communication is of utmost importance to the progress of scientific research. The speed of such development, however, may be affected by the scarcity of reviewers to referee the quality of research articles. In this context, automatic approaches that are able to query linguistic segments in written contributions by detecting the presence or absence of common rhetorical patterns have become a necessity in the refereeing process. This paper aims to compare supervised machine learning techniques tested to accomplish genre analysis in Introduction sections of software engineering articles. A semi-supervised approach to augment the number of annotated sentences in SciSents¹ was performed. Two supervised approaches using SVM and logistic regression to assess the F-score for genre analysis in the corpus were undertaken. A technique based on logistic regression and BERT has been found to perform genre analysis highly satisfactorily with an average of 88.25 on F-score when retrieving patterns at an overall level.

1 Introduction

Written communication plays a fundamental role in scholarly development. Evidence for this is the high number of estimated publications and journals (Larsen and von Ins, 2010; Björk et al., 2008; Mabe, 2003). In this scenario, the reviewing process is a crucial pathway for improving publication quality, as it acts as a filter through which suitable research papers are selected for publication (Ware and Mabe, 2015). In principle, although academic gatekeeping does not entail rigid language rubrics, scientific publications on the whole present standardised conventions such as preference for passive constructions, high nominal style, paper division in sections, and use of lexical and phrasal structures to indicate the function and purpose of each text portion (Seaghdha and Teufel, 2014). Disseminated linguistic work aiming to systematically describe writing organization with a focus on the Introduction section is the CARS (Create a Research Space) model (Swales, 1990). CARS approaches genre analysis by introducing two concepts, namely Moves and Steps. Whereas a Move represents the objectives and functions of a text segment at an overall level, a Step further elaborates on explaining how the rhetorical means are specifically used to perform the function of a Move (Ruiying and Allison, 2003) (see examples in Table 1). Despite models serving as a basis for the reviewing process (e.g. CARS), the availability of reviewers to evaluate scientific publications does not keep pace with the ever-growing number of papers which require gatekeeping (Fox, 2017), therefore making computational techniques necessary.

Computational approaches may be implemented to query linguistic segments automatically in research articles by indicating the presence or absence of commonly used rhetorical patterns. Automatic approaches such as Support Vector Machines (SVM) (Bennett and Demiriz, 1999; Tang et al., 2007) can be employed to perform genre analysis due to its productive results regarding textual issues (Horn et al., 2014; Fernández-Delgado et al., 2014). Nonetheless, they require annotated data, which are scant in the literature and not easily obtained, with the existing ones having limited amount of input (Fisas et al., 2015, 2016; Seaghdha and Teufel, 2014; Anthony and Lashkia, 2003; Pendar and Cotos, 2008; Cotos and Pendar, 2016; Fiacco et al., 2019). Manual annotation is an arduous, expensive and time-consuming task as it requires expert human annotators. To tackle this issue, SVM may be used as a semi-supervised approach, in which considerable amounts of labeled and unlabeled data are
utilized together to form more solid classifiers (Zhu, 2005). In this context, this work aims to evaluate supervised and semi-supervised machine learning techniques for automatic retrieval of rhetorical patterns within Swales’ CARS genre analysis schema in research articles. This investigation was carried out into SciSents corpus and, for this reason, is restricted to the Introduction section of software engineering articles. This paper has two main objectives: the first is to augment the number of annotations in SciSents corpus, and the second is to compare and assess F-Scores generated by supervised approaches for genre analysis. For such, we performed a comparison between SVMs and logistic regression techniques for the classification task. For sentence encoding, we evaluated novel approaches such as Universal Sentence Encoder (Cer et al., 2018a) and BERT (Devlin et al., 2019).

This paper proceeds as follows: firstly, we review state-of-the-art works on genre analysis automation. Next, we present and describe the corpus and the semi-supervised annotation procedures. Also, we comparatively discuss the main features of the techniques employed in the experiments. We then address the included implementation details and, finally, report the results.

2 Related Work

One relevant reference for genre analysis automation is the work of Anthony and Lashkia (2003), who proposed a computer software tool for outlining a research article’s structure. Based on Swales’ schema, the tool (named Mover) aimed at presenting to learners a panorama of the move structure utilized in RA. The tool scanned 100 information technology articles abstracts comprising 692 sentences. The abstracts were manually annotated on the grounds of a Modified Create a Research Space (CARS) model proposed by Anthony (1999). The model includes Swales’ (1990) 3 Moves, as well as 12 Steps. Since this is a general and small-sized model designed for the Introduction section, not all Steps appeared in the dataset. A modified bag of words was utilized to represent the text so it could be machine manipulated. In a traditional bag of words, dataset sentences would be tokenized in single words. However, the authors added clusters of sequential words in order to allow the system to operate at the discourse level, therefore naming the model as Bag of Clusters. As well as allowing the system to identify steps only possible to classify if preceding or subsequent Steps are known, an additional “location” feature was added to the bag of clusters model. The model’s output fed a Naive Bayes classifier which performed consistently with an average Step accuracy rate of 68% (ranging from 17% - Indicate gap - to 92% - Announce research). The authors justified the poor results by the scarce training items from these Steps. Through error analysis, they observed that when the software presented flaws, the incorrectly categorized Step tended to fall within the same Move. In order to improve accuracy the two most probable classifications were used in a second experiment. In this turn, the user had to select the most appropriate option. With this procedure, accuracy achieved 86%. However, despite the productive result, the reduced number of articles and sentences was a hindrance for further validation.

Pendar and Cotos (2008) attempted to devise a pedagogical tool for automating discourse evaluation. The purpose was to appraise academic writing drafts in agreement with an adapted model based on CARS, to compare it with other papers from the same discipline and to provide feedback to the student. To develop such a tool, a text-categorization approach using Support Vector Machine (SVM) for sentence classification in research article introductions drawn on Swales’ rhetorical moves was employed. An experiment was conducted with a corpus named Intelligent Academic Discourse Evaluator (IADE) consisting of 11,149 sentences from 401 Introduction sections in 20 academic disciplines. Each sentence was manually annotated within the Moves from CARS schema. To execute the classification, sentences were stemmed and represented in an n-dimensional vector (up to word trigrams). Experiment results were encouraging (with an accuracy above 70%), but the dataset was relatively small and did not take Steps into account.

Cotos and Pendar (2016) made progress in their own 2008 work by increasing IADE’s size to 1,020 research articles across 51 disciplines. Sentences were also annotated according to the CARS model, but this time including both Moves and Steps. An SVM classifier with the previous settings achieved a Move accuracy of 72.6% and a Step accuracy of 72.9%.

Fiacco et al. (2019) presented a neural network architecture composed of a Bi-LSTM with CRF
as an automated approach to examine rhetorical structure in student writing. The embedding layer was initialised with a pre-trained representation of GloVe (Pennington et al., 2014) and was fine-tuned to the dataset to produce more accurate word representation. Two datasets were used to test the model: IADE (Pendar and Cotos, 2008; Cotos and Pendar, 2016) and Research Writing Tutor (RWT) comprising 900 full research articles (not only Introduction sections) across 30 academic disciplines. RWT was manually annotated and sentences with one communicative goal and more than one functional strategy could be labeled with several steps; a sentence could be assigned with a secondary Move/Step tag if it had more than one communicative goal. Experiments results achieved a precision and recall of 77%, and an F1-score of 76% for the classification task in RWT dataset.

Due to the data paucity problem present in the aforementioned works, this study proposes a semi-supervised approach as a contribution towards genre analysis automation as far as the CARS framework is concerned. A detailed explanation of the procedures can be found in the following sections.

3 Semi-Supervised Approach

3.1 Data

We used SciSents, a dataset of software engineering research article sentences. This data resource is based on 9,193 software engineering articles published between the years 2000 and 2018 in highly-cited journals and conference proceedings. The corpus consists of 322,630 sentences from Introduction sections. From this amount we randomly extracted 595 sentences as our dataset, which was then manually annotated across 13 Steps within 3 Moves as shown in Table 1.

3.2 Models

We automatically performed the genre analysis classification task comparing SVMs and logistic regression as classifiers as well as BERT (Devlin et al., 2019) and Universal Sentence Encoder (Cer et al., 2018a) as sentence embeddings.

3.2.1 Classifiers

SVM: Support Vector Machines are non-parametric and deterministic algorithms based on statistical learning. They have been used specially in NLP (Joachims, 1998; Yang, 1999; Goudjil et al., 2018). SVM builds a hyperplane in a multi-dimensional space with the aim of training a set of labeled instances which create a boundary between distinct classes (Hearst et al., 1998; Joachims, 1998).

Logistic Regression: Logistic regression is a statistical technique for binary classification that can also be applied to multi-class classification by treating genre analysis issues as a binary classification problem (Ifrim et al., 2008). It computes probabilities of classes using a logistic function and then constructs a linear hyperplane separating those classes.

3.2.2 Features

Universal Sentence Encoder: Universal sentence encoding (Cer et al., 2018a) generates embedding vectors by encoding greater-than-word length text using two models: transformer architecture (Vaswani et al., 2017) and Deep Averaging Network (DAN) (Iyyer et al., 2015). Transformer architecture encoder consumes substantial resources and imposes complexity to the model aiming at high accuracy. It is context-aware and takes into account the ordering and the identity of all words in context. It also uses attention to compute the representations of words in a sentence. The second encoding model (i.e. DAN) assumes lightly reduced accuracy aiming at efficient inference. It receives embeddings for words and bi-grams as input, computes its average and inserts it into a feedforward Deep Neural Network (DNN) to create sentence embeddings. The output of both models is a 512-dimensional sentence embedding.

BERT: Bidirectional Encoder Representations from Transformers or BERT (Devlin et al., 2019) is a masked-language model for representing text and comprises a multi-layered bidirectional transformer encoder used for pre-training on a large unlabeled text corpus. It aims at modelling masked-language as well as predicting the next sentence. A random sample of the tokens is masked (replaced with the special token), the next sentence is predicted and BERT proceeds with training and optimization until it obtains satisfactory results (Liu et al., 2019).

3.3 Method

3.3.1 Semi Supervision

To increase the number of annotated sentences in SciSents, we employed a semi supervised strategy, training an SVM in the labeled part of corpus to
classify the unlabeled part. For such, the corpus phrases were represented in a 1024-position vector using BERT (Devlin et al., 2019), following the implementation of Xiao (2018) as described in section 3.2.2.

Following the annotation stage, the probability of the corpus sentences falling into each of the 13 Steps in SciSents was computed. The 20 most likely sentences for each Step (260 in total) were manually checked by a human linguistic expert with considerable knowledge on Swales’ CARS model. Through this analysis we identified 228 correctly classified sentences against 5 wrongly classified ones. 27 sentences could not be categorized because of a few tokenization glitches. At the end of this stage, 233 sentences were added to the annotated set (including the former 5 incorrectly classified ones which were later corrected), amounting to a total of 828 manually annotated sentences. A new SVM training was then administered with this annotated set.

A second round of semi-supervised annotation followed, in which the linguistic expert analysed random sentences with different probabilities for Steps calculated by the second SVM training. A total of 481 random sentences were manually checked, of which 308 were marked as correctly classified and 173 marked as incorrectly classified. The misclassified sentences were manually reclassified so they could be added to the correct ones within the annotated set. Sentences with tokenization problems were discarded. At the end of this stage, 1,309 sentences were part of the manually annotated set (see Table 1). This corpus was used in the experiments, which are presented in the following section.

### 3.3.2 Evaluation

We used three measures to assess model performance: Precision, Recall, and F-score. Precision measures the proportion of correctly classified sentences out of the total number of annotated sentences, while Recall estimates the proportion of correctly annotated sentences out of the incorrectly predicted sentences plus the correctly classified sentences. F-Score in turn is the harmonic mean of both Precision and Recall (Goutte and Gaussier, 2005). Each technique was trained using 5 fold cross-validation and averages across F-Score results on test folds were reported.

Two embeddings were generated for the experiments. The first consisted of generating corpus sentence representation individually and the second

<table>
<thead>
<tr>
<th>Move</th>
<th>Step</th>
<th>SS</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Establishing the territory</strong></td>
<td>187</td>
<td>257</td>
<td>444</td>
<td></td>
</tr>
<tr>
<td>M1-S01 - Establishing the importance of the topic for the discipline</td>
<td>37</td>
<td>57</td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>M1-S02 - Establishing the importance of the topic for the world or society</td>
<td>45</td>
<td>65</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td>M1-S03 - Establishing the importance of the topic as a problem to be addressed</td>
<td>45</td>
<td>63</td>
<td>116</td>
<td></td>
</tr>
<tr>
<td>M1-S04 - Referring to previous work to establish what is already known</td>
<td>60</td>
<td>72</td>
<td>136</td>
<td></td>
</tr>
<tr>
<td><strong>Establishing a niche</strong></td>
<td>98</td>
<td>136</td>
<td>199</td>
<td></td>
</tr>
<tr>
<td>M2-S05 - Identifying and highlighting inadequacies, weaknesses, controversies and negative outcomes within the field of study</td>
<td>45</td>
<td>63</td>
<td>113</td>
<td></td>
</tr>
<tr>
<td>M2-S06 - Identifying a knowledge gap, a lack of or paucity of previous research in the field of study</td>
<td>53</td>
<td>73</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td><strong>Occupying the niche</strong></td>
<td>310</td>
<td>435</td>
<td>666</td>
<td></td>
</tr>
<tr>
<td>M3-S07 - Stating the focus, aim, purpose or argument of the current research</td>
<td>44</td>
<td>64</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>M3-S08 - Setting out the research questions or hypotheses</td>
<td>36</td>
<td>56</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>M3-S09 - Describing the research design and the methods used</td>
<td>47</td>
<td>67</td>
<td>122</td>
<td></td>
</tr>
<tr>
<td>M3-S10 - Explaining the significance or give reasons for personal interest in the current study</td>
<td>33</td>
<td>42</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>M3-S11 - Describing the limitations of the current study</td>
<td>31</td>
<td>50</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>M3-S12 - Outlining the structure of a chapter, paper, thesis or dissertation</td>
<td>80</td>
<td>99</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td>M3-S13 - Explaining Keywords (also refer to Defining Terms)</td>
<td>39</td>
<td>57</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>595</td>
<td>828</td>
<td>1,309</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Number of manually annotated sentences by Move and by Step in SciSents (SS), in semi supervised Round 1 (R1) and Round 2 (R2).
a representation in co-occurrence with the previous sentence. The purpose of this second approach was to investigate whether the previous sentence had an influence on the subsequent one in terms of genre analysis. In cases where the sentence was not preceded by any other, representation was calculated with that sentence solely. We highlight that the previous sentences were not necessarily the immediately preceding ones since invalid sentences were removed from SciSents during the preprocessing stage. The embedded sentences and phrase labels were the input for SVM training.

3.4 Baselines

The SVM-BERT pair, availed as the basis for the semi-supervised annotation (considering the embeddings generated from individual corpus sentences), was used for comparison with the rest of the experiment. For each technique, we explored two combinations of sentence embedding features: Universal Sentence Encoder and BERT. As to the former, a TensorFlow implementation\(^3\) (Cer et al., 2018b) was used and generated a 512-dimensional sentence embedding vector. Regarding the latter, BERT as a Service (Xiao, 2018) was employed and generated a 1024-position vector.

4 Results

We report the F-score averaged over the folds of our techniques in Tables 2, 3, 4 and 5. Each table column shows the result of an experiment type comprising a technique (SVM or logistic regression), a sentence embedding technique (BERT or universal sentence encoder), and an annotated set (SciSents, semi-supervised Round 1 and semi-supervised Round 2).

Table 2 summarizes the results of experiments on the Steps categories when using one sentence alone to generate the embeddings\(^4\). Except for 2 Steps (M1-S03- Establishing the importance of the topic as a problem to be addressed and M3-S11- Describing the limitations of the current study), logistic regression technique with BERT presented higher scores overall. In 6 times out of these the highest results in the semi-supervised annotation Round 2 were achieved for the following Steps: M1-S01-Establishing the importance of the topic for the discipline; M1-S02-Establishing the importance of the topic for the world or society; M1-S04-Referring to previous work to establish what is already known; M2-S05-Identifying and highlighting inadequacies, weaknesses, controversies and negative outcomes within the field of study; M3-S09-Describing the research design and the methods used; M3-S10-Explaining the significance or give reasons for personal interest in the current study. In the remaining 5 Steps (i.e. M2-S06-Identifying a knowledge gap, a lack of or paucity of previous research in the field of study; M3-S07-Stating the focus, aim, purpose or argument of the current research; M3-S08-Setting out the research questions or hypotheses; M3-S12-Outlining the structure of a chapter, paper, thesis or dissertation; M3-S13-Explaining Keywords (also refer to Defining Terms)), better scores were obtained in the semi-supervised annotation Round 1.

The best performance among all results was achieved for Step M3-S12 (Outlining the structure of a chapter, paper, thesis or dissertation) in semi-supervised annotation Round 2 using logistic regression and BERT, which showed a 0.8856 F-Score. The results in Table 2 for M3-S12 were higher than 0.84. This result can be explained by the fact that sentences within this Step are prototypical (e.g. “The paper is structured as follows”, “Finally, Section 6 concludes the paper and discusses its implications.”, and “The remainder of this paper begins with a comparison to related work (Section 2), followed by an overview of the approach used to create a corpus, perform change classification, and evaluate its performance (Section 3).”).

The worst performance among all results in Table 2 was a 0.1152 F-Score produced in M3-S10 (Explaining the significance or give reasons for personal interest in the current study) when using logistic regression and universal sentence encoder in SciSents annotated sentences. One possible explanation for this low performance is that the number of annotations is one of the smallest among all Steps (33 sentences). In addition, this result can be justified by the fact that sentence type used in this Step is quite varied such as “Our experiments, backed by a human study, suggest DeltaDoc could replace over 89% of human-generated What log messages.”, “This combines visualizations, providing a high level overview, and wiki pages, providing detailed information juxtaposed in a focus-plus-context oriented format.”, and “The backward analysis computes an over approxima-
tion of all possible inputs that can generate those attack strings." Throughout annotations rounds, M3-S10 improved its results and reached a performance of 0.4092. The best performance in Table 2 for M3-S10 scored 0.5081 when using the SVM-BERT pair.

Table 3 shows the performance of the experiments on Steps when using both actual and previous sentences to generate vector representation. The pair logistic regression with BERT surpassed other pairs in 7 (M1-S02, M1-S03, M2-S05, M2-S06, M3-S08, M3-S09, and M3-S10) out of the 13 Steps. As to the results regarding sole sentence embedding, the best performance among all was achieved in M3-S12 but this time in SciSents annotations using SVM and BERT with a 0.8932 F-Score. One of the reasons that may have contributed to this result even before semi-supervised rounds is the annotated sentence number (80) being the highest among all Steps. The worst performance in this type of experiment was a 0.1152 F-Score output for M3-S10.

We notice that results shown in Table 2 are more productive than the ones from Table 3 in 44 (or 56.41%) out of 78 when considering experiments that used BERT in isolation. When analysing only the best scores for each Step, Table 2 presents best results in 8 (61.53%), whereas Table 3 shows the most productive scores in 4 (30.77%) out of 13 cases. There was a draw in one case. Performance with universal sentence encoding was the same on both tables.

Table 4 summarizes the results of experiments on Moves when using one sentence solely to generate the embeddings. The best F-score for each Move was achieved with logistic regression and BERT in semi-supervised annotation Round 1 with an average of 0.8569 against an average of 0.8422 for Round 2. The lowest score in Round 2 was 0.7867 for M1 (Establishing the territory) whereas M2 (Establishing a niche) scored 0.8564. M3 (Occupying the niche) outperformed all other results with a score of 0.9275. When we compare these results with their respective scores in semi-supervised annotation (Round 2), there is a difference of 0.0126, 0.0129, and of 0.0187 between Moves M1, M2 and M3 respectively.

Table 5 presents results on Moves when the vector representation is created using the actual sentence in conjunction with the previous sentence. Similar to the technique with sole sentence embeddings for Moves, the best F-Score was reached with logistic regression and BERT. But this time M1 and M2 were reached in semi-supervised annotation in Round 2 and M3 in semi-supervised annotation in Round 1. When we compare scores from Table 4 with those from Table 5 we can notice that the figures on the former surpass all respective results on the latter when considering BERT alone. Again, when Universal Sentence Encoder was used there was no difference between the embedding from one sentence alone and from a sentence co-occurring with its previous one.

5 Discussion

The present study was designed to augment the number of annotations in SciSents corpus and to compare results in supervised machine learning techniques for genre analysis in software engineering research articles. The number of annotated sentences was increased from the 595 ones in SciSents to 1309 through two semi-supervised rounds using SVM.

SVM versus Logistic Regression: Logistic regression produced higher outcomes than SVM in 64% of the experiments. When associated with BERT, logistic regression beats SVM in 85% of cases, but when in conjunction with USE, SVM outperformed logistic regression in 57% of experiments.

Universal Sentence Encoder versus BERT: Vector representation provided by BERT delivered higher scores than Universal Sentence Encoder did in 75.5% of the tested sets. When BERT was employed with logistic regression, the results overcame other experiments in 81% of cases. Thus, from the pairs of techniques tested, the indicated one for genre analysis is logistic regression with BERT.

Vector representation - sentence alone versus co-occurring sentences: One finding in the experiments in supervised machine learning techniques is that, in most cases, the use of sentence embedding generated from the sentence alone provided more productive results than those with the use of the actual sentence together with its preceding one.

---

5The strongest F-score in each row is in bold.
6The strongest F-score in each row is in bold.
7The strongest F-score in each row is in bold.
<table>
<thead>
<tr>
<th>Step</th>
<th>SVM-BERT</th>
<th>SVM-USE</th>
<th>LR-BERT</th>
<th>LR-USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM-BERT</td>
<td>SVM-USE</td>
<td>LR-BERT</td>
<td>LR-USE</td>
<td></td>
</tr>
<tr>
<td>M1-S01</td>
<td>34.00</td>
<td>53.28</td>
<td>65.05</td>
<td></td>
</tr>
<tr>
<td>M1-S02</td>
<td>31.87</td>
<td>52.10</td>
<td>61.56</td>
<td></td>
</tr>
<tr>
<td>M1-S03</td>
<td>47.51</td>
<td>62.80</td>
<td>63.54</td>
<td></td>
</tr>
<tr>
<td>M1-S04</td>
<td>33.14</td>
<td>44.57</td>
<td>54.47</td>
<td></td>
</tr>
<tr>
<td>M2-S05</td>
<td>41.8</td>
<td>50.43</td>
<td>55.88</td>
<td></td>
</tr>
<tr>
<td>M2-S06</td>
<td>69.09</td>
<td>80.46</td>
<td>78.00</td>
<td></td>
</tr>
<tr>
<td>M3-S07</td>
<td>50.75</td>
<td>68.65</td>
<td>70.30</td>
<td></td>
</tr>
<tr>
<td>M3-S08</td>
<td>58.56</td>
<td>75.33</td>
<td>63.19</td>
<td></td>
</tr>
<tr>
<td>M3-S09</td>
<td>27.92</td>
<td>46.02</td>
<td>63.38</td>
<td></td>
</tr>
<tr>
<td>M3-S10</td>
<td>31.98</td>
<td>34.50</td>
<td>50.41</td>
<td></td>
</tr>
<tr>
<td>M3-S11</td>
<td>78.97</td>
<td>86.12</td>
<td>76.75</td>
<td></td>
</tr>
<tr>
<td>M3-S12</td>
<td>82.94</td>
<td>85.71</td>
<td>85.37</td>
<td></td>
</tr>
<tr>
<td>M3-S13</td>
<td>81.69</td>
<td>85.42</td>
<td>81.45</td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>53.93</td>
<td>63.84</td>
<td>66.71</td>
<td>52.27 62.46 60.68</td>
</tr>
<tr>
<td>SVM-BERT</td>
<td>SVM-USE</td>
<td>LR-BERT</td>
<td>LR-USE</td>
<td></td>
</tr>
<tr>
<td>M1-S01</td>
<td>27.22</td>
<td>47.55</td>
<td>56.63</td>
<td></td>
</tr>
<tr>
<td>M1-S02</td>
<td>60.5</td>
<td>64.50</td>
<td>70.50</td>
<td></td>
</tr>
<tr>
<td>M1-S03</td>
<td>41.16</td>
<td>53.64</td>
<td>58.38</td>
<td></td>
</tr>
<tr>
<td>M1-S04</td>
<td>44.42</td>
<td>52.09</td>
<td>60.84</td>
<td></td>
</tr>
<tr>
<td>M2-S05</td>
<td>47.59</td>
<td>52.68</td>
<td>61.10</td>
<td></td>
</tr>
<tr>
<td>M2-S06</td>
<td>56.02</td>
<td>74.03</td>
<td>71.44</td>
<td></td>
</tr>
<tr>
<td>M3-S07</td>
<td>29.63</td>
<td>55.37</td>
<td>53.37</td>
<td></td>
</tr>
<tr>
<td>M3-S08</td>
<td>47.48</td>
<td>64.74</td>
<td>52.68</td>
<td></td>
</tr>
<tr>
<td>M3-S09</td>
<td>40.29</td>
<td>52.12</td>
<td>61.83</td>
<td></td>
</tr>
<tr>
<td>M3-S10</td>
<td>50.5</td>
<td>39.77</td>
<td>56.64</td>
<td></td>
</tr>
<tr>
<td>M3-S11</td>
<td>69.51</td>
<td>72.71</td>
<td>71.02</td>
<td></td>
</tr>
<tr>
<td>M3-S12</td>
<td>89.31</td>
<td>87.84</td>
<td>84.65</td>
<td></td>
</tr>
<tr>
<td>M3-S13</td>
<td>78.60</td>
<td>77.56</td>
<td>82.51</td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>55.37</td>
<td>63.72</td>
<td>65.37</td>
<td>52.27 62.46 60.68</td>
</tr>
</tbody>
</table>

Table 2: Experiment results per Step on SciSents (SS), semi supervised Round 1 (R1) and Round 2 (R2) annotated sentence sets using one sentence solely to create vector representation (LR = Logistic Regression; USE = Universal Sentence Encoder).

<table>
<thead>
<tr>
<th>Step</th>
<th>SVM-BERT</th>
<th>SVM-USE</th>
<th>LR-BERT</th>
<th>LR-USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM-BERT</td>
<td>SVM-USE</td>
<td>LR-BERT</td>
<td>LR-USE</td>
<td></td>
</tr>
<tr>
<td>M1-S01</td>
<td>27.22</td>
<td>47.55</td>
<td>56.63</td>
<td></td>
</tr>
<tr>
<td>M1-S02</td>
<td>60.5</td>
<td>64.50</td>
<td>70.50</td>
<td></td>
</tr>
<tr>
<td>M1-S03</td>
<td>41.16</td>
<td>53.64</td>
<td>58.38</td>
<td></td>
</tr>
<tr>
<td>M1-S04</td>
<td>44.42</td>
<td>52.09</td>
<td>60.84</td>
<td></td>
</tr>
<tr>
<td>M2-S05</td>
<td>47.59</td>
<td>52.68</td>
<td>61.10</td>
<td></td>
</tr>
<tr>
<td>M2-S06</td>
<td>56.02</td>
<td>74.03</td>
<td>71.44</td>
<td></td>
</tr>
<tr>
<td>M3-S07</td>
<td>29.63</td>
<td>55.37</td>
<td>53.37</td>
<td></td>
</tr>
<tr>
<td>M3-S08</td>
<td>47.48</td>
<td>64.74</td>
<td>52.68</td>
<td></td>
</tr>
<tr>
<td>M3-S09</td>
<td>40.29</td>
<td>52.12</td>
<td>61.83</td>
<td></td>
</tr>
<tr>
<td>M3-S10</td>
<td>50.5</td>
<td>39.77</td>
<td>56.64</td>
<td></td>
</tr>
<tr>
<td>M3-S11</td>
<td>69.51</td>
<td>72.71</td>
<td>71.02</td>
<td></td>
</tr>
<tr>
<td>M3-S12</td>
<td>89.31</td>
<td>87.84</td>
<td>84.65</td>
<td></td>
</tr>
<tr>
<td>M3-S13</td>
<td>78.60</td>
<td>77.56</td>
<td>82.51</td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>55.37</td>
<td>63.72</td>
<td>65.37</td>
<td>52.27 62.46 60.68</td>
</tr>
</tbody>
</table>

Table 3: Experiment results by Steps on SciSents (SS), semi supervised Round 1 (R1) and Round 2 (R2) annotated sentence sets using a sentence in co-occurrence with its immediately preceding one to create vector representation (LR = Logistic Regression; USE = Universal Sentence Encoder).

<table>
<thead>
<tr>
<th>Move</th>
<th>SVM-BERT</th>
<th>SVM-USE</th>
<th>LR-BERT</th>
<th>LR-USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM-BERT</td>
<td>SVM-USE</td>
<td>LR-BERT</td>
<td>LR-USE</td>
<td></td>
</tr>
<tr>
<td>M1</td>
<td>72.5</td>
<td>75.91</td>
<td>72.68</td>
<td></td>
</tr>
<tr>
<td>M2</td>
<td>79.3</td>
<td>84.01</td>
<td>79.71</td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>82.78</td>
<td>86.86</td>
<td>82.98</td>
<td>78.57 82.68 79.81</td>
</tr>
</tbody>
</table>

Table 4: Experiment results by Moves on SciSents (SS), semi supervised Round 1 (R1) and Round 2 (R2) annotated sentence sets using one sentence solely to create vector representation (LR = Logistic Regression; USE = Universal Sentence Encoder).

<table>
<thead>
<tr>
<th>Move</th>
<th>SVM-BERT</th>
<th>SVM-USE</th>
<th>LR-BERT</th>
<th>LR-USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM-BERT</td>
<td>SVM-USE</td>
<td>LR-BERT</td>
<td>LR-USE</td>
<td></td>
</tr>
<tr>
<td>M1</td>
<td>72.5</td>
<td>75.91</td>
<td>72.68</td>
<td></td>
</tr>
<tr>
<td>M2</td>
<td>79.3</td>
<td>84.01</td>
<td>79.71</td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>82.78</td>
<td>86.86</td>
<td>82.98</td>
<td>78.57 82.68 79.81</td>
</tr>
</tbody>
</table>

Table 5: Experiment results by Move on SciSents (SS), semi supervised Round 1 (R1) and Round 2 (R2) annotated sentence sets using the sentence co-occurring with its previous one to create the vector representation (LR = Logistic Regression; USE = Universal Sentence Encoder).
Semi-Supervised Approach: When analysing the evolution of results throughout the annotation process within each experiment type we can notice that they did not always improve accordingly. In Table 2, when we compare SciSents with annotations from Round 1, there was no increase in the F-score (despite one situation only representing 1.92% out of the total). When comparing annotations from Round 1 with Round 2, the latter outperformed the former in 46.15% of the cases. A possible explanation is that in Round 1 highest-ranked sentences by SVM were annotated while in Round 2 sentences with random probabilities were annotated. Thus, in Round 1, similar sentences to those the techniques already knew were included, whereas in Round 2 sentences which were different from those the techniques knew (but still fell into that Step) were included.

When approaching annotation evolution throughout Table 3 we observe that experiments within Round 1 annotations outperformed experiments in SciSents in 90.38% of the results. When comparing experiments in annotations between Round 1 and Round 2 there is a 50% (26 times) draw in which Round 2 showed better results than Round 1. The same analysis in Tables 4 and 5 shows that experiments in Round 1 annotations outperformed experiments in SciSents. When comparing annotations between Round 1 and Round 2 we observe no improvement in the latter (as shown in Table 4), but some improvement in 33.33% of the overall cases, as we see in Table 5. These results indicate that the second round of annotation may have included sentence types unknown to the technique.

Although most of the best results for Steps were achieved with a second round annotation set, setbacks were also present, thus indicating the need for more annotations for probabilities potentially corresponding to the categories set for Steps. These annotations may also contribute to genre analysis regarding Moves, despite results presenting insignificant improvements with a second round of annotations.

6 Conclusion

The present study compared supervised machine learning techniques which automatically retrieved linguistic segments from research articles. Firstly, we used a semi-supervised approach to increase the number of annotated sentences in SciSents corpus. Next, we used two supervised and two sentence embedding techniques to carry out genre analysis on the dataset. The results suggest that an approach based on logistic regression and BERT presents higher scores for genre analysis. In addition, although a semi-supervised annotation process has proven to contribute to the overall procedure, it lacks elements with random probabilities for substantial improvement.

As future work, the semi-supervised annotation process and the techniques hereby described can be used for annotating other sections of software engineering research articles. Also, the same analyses could be applied to articles from other domains so that cross-disciplinary rhetorical differences could be identified.

References


Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of


Peder Olesen Larsen and Markus von Ins. 2010. The rate of growth in scientific publication and the decline in coverage provided by Science Citation Index. Scientometrics, 84(3):575–603.


Han Xiao. 2018. bert-as-service.


Enriching the Transformer with Linguistic Factors for Low-Resource Machine Translation

Jordi Armengol-Estapé, Marta R. Costa-jussà, Carlos Escolano
TALP Research Center, Universitat Politècnica de Catalunya, Barcelona
jordi.armengol.estape@gmail.com, {marta.ruiz, carlos.escolano}@upc.edu

Abstract
Introducing factors, that is to say, word features such as linguistic information referring to the source tokens, is known to improve the results of neural machine translation systems in certain settings, typically in recurrent architectures. This study proposes enhancing the current state-of-the-art neural machine translation architecture, the Transformer, so that it allows to introduce external knowledge. In particular, our proposed modification, the Factored Transformer, uses linguistic factors that insert additional knowledge into the machine translation system. Apart from using different kinds of features, we study the effect of different architectural configurations. Specifically, we analyze the performance of combining words and features at the embedding level or at the encoder level, and we experiment with two different combination strategies. With the best-found configuration, we show improvements of 0.8 BLEU over the baseline Transformer in the IWSLT German-to-English task. Moreover, we experiment with the more challenging FLoRes English-to-Nepali benchmark, which includes both extremely low-resourced and very distant languages, and obtain an improvement of 1.2 BLEU.

1 Introduction
Many classical Natural Language Processing (NLP) pipelines used linguistic features (Koehn and Hoang, 2007; Du et al., 2016). In recent years, the rise of neural architectures has diminished the importance of the aforementioned features. Nevertheless, some works have still shown the effectiveness of introducing linguistic information into neural machine translation systems, typically in recurrent sequence-to-sequence (Seq2seq) architectures (Sennrich and Haddow, 2016; García-Martínez et al., 2016; España-Bonet and van Genabith, 2018). By factored Neural Machine Translation (NMT), we refer to the use of word features alongside the words themselves to improve translation quality. Both the encoder and the decoder of a Seq2seq architecture can be modified to obtain better translations (García-Martínez et al., 2016). The most prominent approach consists of modifying the encoder such that instead of only one embedding layer, the encoder has as many embedding layers as factors, one for words themselves and one for each feature, and then the embedding vectors are concatenated and input to the rest of the model, which remains unchanged (Sennrich and Haddow, 2016). The embedding sizes are set according to the respective vocabularies of the features. Note that they used Byte Pair Encoding (BPE) (Sennrich et al., 2016), an unsupervised preprocessing step for automatically splitting words into subwords with the goal of improving the translation of rare or unseen words. Thus, the features had to be repeated for each subword.

In España-Bonet and van Genabith (2018), the exact same architecture was used, except that this new proposal used concepts extracted from linked data database, BabelNet (Navigli and Ponzetto, 2012). These semantic features, synsets, were shown to improve zero-shot translations. All the cited works obtained moderate improvements with respect to the BLEU scores of the corresponding baselines.

Some works have previously proposed additional ways to combine sources and introduce hierarchical linguistic information (Currey and Heafield, 2019, 2018; Libovický et al., 2018; Tebbifakhr et al., 2018).

The main goal of this work, and differently from previous works using NMT architectures based on recurrent neural networks, is to modify the Transformer to make it compatible with factored NMT with an architecture that we call Factored Transformer and inject linguistic knowledge and concepts extracted from linked data, BabelNet (Nav-
igli and Ponzetto, 2012). We focus on low-resource datasets.

2 Factored Transformer

Unlike the vanilla Transformer (Vaswani et al., 2017), the Factored Transformer can work with factors; that is, instead of just being input the original source sequence, it can work with an arbitrary number of feature sequences. Those features can be injected at embedding-level, as in the previous works we described above (but in a Transformer instead of a recurrent-based seq2seq architecture), or at the encoder level.

1-encoder model (depicted in Figure 1, top): Each factor, including the words themselves, has its own embedding layer. The embedding vectors of the different factors are combined, positional encoding is summed and input to the following layer. The rest of the model remains unchanged. The positional encoding is summed to the combined vector and not to each individual embedding because we are not modifying the length of the sequence; therefore, the relative positions remain unchanged.

N-encoders model (depicted in Figure 1, bottom): We intuited that features with large vocabulary sizes could benefit from having a specific encoder. In this variant, each factor has its own full encoder (instead of just its own embedding layer). The outputs from the encoder are combined and input to the following layer. The rest of the model remains unchanged.

Once we have the outputs of the multiple embedding layers (the 1-encoder) or the N-encoders, they must be aggregated before being input to the next layer. We have considered two combination strategies:

Concatenation: The outputs of the different embedding layers or encoders are concatenated.

Summation: The outputs of the different embedding layers or encoders are summed.

In both cases, the dimensions must agree. The decoder embedding size must be equal to the encoder embedding size. If the outputs from the different encoders or embedding layers are concatenated, they do not need to have the same embedding size, but the resulting embedding size is increased. Instead, if they are summed, they must share the same dimensionality, but the resulting vector size is not increased.

Figure 1: 1-encoder and N-encoders models.
### Table 1: BLEU results. In bold, best results.

<table>
<thead>
<tr>
<th>MODEL</th>
<th>COMB.</th>
<th>FEATURE</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>-</td>
<td>-</td>
<td>34.08</td>
</tr>
<tr>
<td>Lemmas</td>
<td>-</td>
<td>-</td>
<td>29.83</td>
</tr>
<tr>
<td>1-encoder</td>
<td>Sum</td>
<td>Lemmas</td>
<td>34.35</td>
</tr>
<tr>
<td>1-encoder</td>
<td>Sum</td>
<td>Babelnet</td>
<td>33.66</td>
</tr>
<tr>
<td>1-encoder</td>
<td>Concat</td>
<td>Lemmas</td>
<td>27.10</td>
</tr>
<tr>
<td>N-encoders</td>
<td>Concat</td>
<td>Lemmas</td>
<td>33.58</td>
</tr>
<tr>
<td>N-encoders</td>
<td>Sum</td>
<td>Lemmas</td>
<td>9.71</td>
</tr>
<tr>
<td>Baseline</td>
<td>-</td>
<td>-</td>
<td>36.67</td>
</tr>
<tr>
<td>1-encoder</td>
<td>Sum</td>
<td>Lemmas</td>
<td>37.46</td>
</tr>
</tbody>
</table>

#### 3 Linguistic Features

An arbitrary number of features can be injected into the Factored Transformer, provided they are aligned with words. As follows we describe how linguistic features were extracted and how they were aligned at the subword level.

**Linguistic tagging with StanfordNLP:** The corpus was tagged with linguistic information, namely lemmas, part-of-speech (PoS), word dependencies and morphological features, using StanfordNLP (Qi et al., 2018), and aligned with respect to the original tokenization.

**Synsets extraction:** BabelNet’s API retrieves all possible synsets (semantic identifiers) that a given token may have. Babelfy (Moro et al., 2014) is a word sense disambiguation service based on BabelNet that retrieves the disambiguated synset for each token depending on the sentence-level context. We split the corpus into chunks such that the daily usage limits of the API were not exceeded and no sentence was split in half (because otherwise Babelfy would have missed the context).

Babelfy returns a list of all the detected synsets with their character offsets, and they must be assigned and aligned to the original tokenization of the corpus. The following step was performed to resolve multiword synset conflicts since in the case of synsets composed of more than one token, Babelfy may retrieve one individual synset for each token and a collective one. We decided to prioritize the synset with the largest number of tokens since it seemed to give the most disambiguated information (e.g. the synset semantic network gives more specific information than the individual synsets semantic and network). For the tokens in the corpus that do not have an assigned synset (e.g. articles or punctuation marks), we assign a backup syntactic feature, namely, part-of-speech.

**Feature Alignment at the Subword-level:** To obtain state-of-the-art results in NMT, subwords (typically, BPE) is usually required. This presents a challenge with regard to word features since they must be aligned with the words themselves. The following alternatives were implemented and experimented with: just repeating the word features for each subword; using the BPE symbol in word features, in the same manner this tag is used in BPE for splitting subwords; and subword tags. This last approach was used in (Semnich and Haddow, 2016) and it consisted of repeating the word features for each subword and introducing a new factor, subword tags, to encode the position of the subword in the original word. The 4 possible tags are: B (beginning of subword), I (intermediate subword), E (end of the subword) and O (the word was not split). This approach is not compatible with the multiencoder architecture.

#### 4 Experimental Framework and Results

**Data:** Experiments were conducted with a pair composed of similar languages, the German-to-English translation direction of the IWSLT14 (Cettolo et al., 2014), which is a low-resource dataset (the training set contains about 160,000 sentences). For cleaning and tokenizing, we use the data preparation script proposed by the authors of Fairseq (Ott et al., 2019). We took the test sets from the corpus released for IWSLT14 and IWSLT16. The former was used to test the best configuration, and the latter was used to see the improvement of this configuration in another set. A joint BPE (ie. German and English share subwords) of 32,000 operations is learned from the training data, with a threshold of 50 occurrences for the vocabulary.

Other experiments were conducted with the English-to-Nepali translation direction of the FLoRes Low Resource MT Benchmark (Guzmán et al., 2019). Although this pair has more sentences than the previous one (564,000 parallel sentences), it is considered to be extremely low-resource and far more challenging because of the lack of similarity between the involved languages. In this case, we
learn a joint BPE of 5000 operations (both with an algorithm based on BPE, sentencepiece (Kudo and Richardson, 2018), as proposed by the FLoRes authors, and with the original BPE algorithm).

**Parameters and Configurations:** In the case of German-to-English, we used the Transformer architecture with the hyperparameters proposed by the Fairseq authors: specifically, 6 layers in the encoder and the decoder, 4 attention heads, embedding sizes of 512 and 1024 for the feedforward expansion size, a dropout of 0.3 and a total batch size of 4000 tokens, with a label smoothing of 0.1. For English-to-Nepali, we used the baseline proposed by the FLoRes authors: specifically, 5 layers in the encoder and the decoder, 2 attention heads, embedding sizes of 512 and 2048 for the feedforward expansion size and a total batch size of 4000 tokens, with a label smoothing of 0.2. In both cases, we used the Transformer architecture with the corresponding parameters we described above as the respective baseline systems, and we introduced the modifications of the Factored Transformer without modifying the rest of the architecture and parameters. As mentioned previously, linguistic features were obtained through StanfordNLP (Qi et al., 2018), except the Babelnet synsets. In the case of the latter, we found that approximately 70% of the tokens in the corpus we used did not have an assigned synset and were therefore assigned PoS.

**Preliminary experiments:** We experimented with BPE alignment strategies (including the approaches from section 4.2), and linguistic features extracted from Stanford tagger (lemmas, part-of-speech, word dependencies, morphological features). The preliminary experiments showed that BPE alignment strategies were not very relevant, so we adopted the alignment with BPE by repeating the word feature. In addition, we found that the most promising linguistic feature was lemmas (Sennrich and Haddow, 2016).

**Reported results:** We report experiments with features (lemmas and synsets), architectures (1-encoder and N-encoders systems), and combination strategies (concatenation and summation). Table 1 shows the performance of the baseline and the baseline architecture but with lemmas instead of the original words. We report how different features (lemmas or BabelNet) compare for a given architecture. Then, for the best feature, lemmas, Table 1 compares different architectures, and it is shown that the best architecture is the 1-encoder with summation. Finally, the best performing system (lemmas with a 1-encoder and summation) is evaluated in another test set, IWSLT16. The selected model is relatively efficient, because it only needs an additional embedding layer.

Once we had found that the 1-encoder Factored Transformer with summation and lemmas was a solid configuration for low-resource settings, we applied this combination the more challenging Facebook Low Resource (FLoRes) MT Benchmark. Specifically, we wanted to compare how this architecture performs against the baseline reported in the original work of this benchmark. The authors report the results before applying backtranslation and with sentencepiece, which is 4.30 BLEU. We reproduced that baseline and we got slightly better results (up to 4.38 BLEU). However, our system is designed to work with BPE, not sentencepiece, which is more challenging to align to features (since subwords coming from different words can be combined into a single token). Table 1 shows that our configuration clearly outperformed the baseline with BPE (almost 40% up), and was very close to the results with sentencepiece.

**Discussion:** The 1-encoder system outperforms the N-encoder one. We hypothesize that the N-encoder architecture does not give good results because a completely disentangled representation for each feature is being learned, and this is not an effective strategy for factored NMT. Therefore, it is better to combine features and words at the embedding level, not at the hidden-state level.

In the case of N-encoder with concatenation, if the linguistic features are not useful if they come from a different encoder, the decoder at least can learn to ignore them. In the case of the N-encoder architecture with sum, since the outputs from different encoders, which are potentially in very different spaces, are summed, it is tough for the decoder to interpret the vectors. In this case the decoder should learn to undo a sum, which is more difficult than just learning to ignore half of the vector (i.e., assigning low weights). In the case of the 1-encoder architecture, summation gives a much more compact representation. Summing lemmas allows the decoder layers to have a dimension of 512 (instead of doubling that, which may overfit).

Regarding the reasons why lemmas outperform synsets, we believe that the problem comes from the fact that a significant proportion of the tokens
do not get a synset. Instead, we can tag all words with lemmas. Besides, the use of synsets (BabelNet) intends to help at disambiguating, but the Transformer is already good at this task (Tang et al., 2018).

5 Conclusions

We have shown that the Transformer can take advantage of linguistic features but not synsets. We conclude the best configuration for the Factored Transformer is the 1-encoder model (with multiple embedding layers) with summation instead of concatenation. For the German-to-English IWSLT task, the best configuration for the Factored Transformer shows an improvement of 0.8 BLEU, and for the extremely low-resourced English-to-Nepali task, the improvement is 1.2 BLEU.

In future work, we suggest adapting the alignment algorithm to sentencepiece by combining features coming from different words into a single feature, provided their respective subwords have been merged into a single token. We suggest investigating whether linguistic features are still useful with backtranslation too.

Acknowledgements

The authors want to specially thank Mercedes García and Cristina España for the insightful discussions. This work is supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 947657).

References


82


A Multi-Pass Sieve Coreference Resolution for Indonesian

Valentina Kania Prameswara Artari, Rahmad Mahendra, Meganingrum Arista Jiwanggi, Adityo Anggraito, Indra Budi
Faculty of Computer Science, Universitas Indonesia
Depok 16424, West Java, Indonesia
valentina.kania@ui.ac.id, rahmad.mahendra@cs.ui.ac.id

Abstract

Coreference resolution is an NLP task to find out whether the set of referring expressions belong to the same concept in discourse. A multi-pass sieve is a deterministic coreference model that implements several layers of sieves, where each sieve takes a pair of correlated mentions from a collection of non-coherent mentions. The multi-pass sieve is based on the principle of high precision, followed by increased recall in each sieve. In this work, we examine the portability of multi-pass sieve coreference resolution model to Indonesian language. We conduct the experiment on 201 Wikipedia documents and multi-pass sieve system yields 72.74% of MUC F-measure and 52.18% of BCUBED F-measure.

1 Background

Many Natural Language Processing (NLP) tasks need to incorporate linguistic comprehension beyond semantics understanding. Coreference resolution is an important discourse-level NLP pipeline that can be used to support a number of NLP applications, such as question answering, summarization, and dialogue system. Coreference resolution task aims to evaluate whether a set of expressions in the text refer to each other, in other words whether they describe the same entity in real-world situation (Hirschman and Chinchor, 1998; Sukthanker et al., 2020)

There are not many coreference resolution studies in Indonesian. Budi et al. (2006) worked on the Indonesian coreference resolution task by applying the Association Rules. Suherik and Purwarianti (2017) developed a coreference resolution system using supervised classifier. They utilized lexical and syntactic features to connects pronouns to named entities, between named entities, and between pronouns.

2 Noun Phrases in Indonesian

The set of referential phenomena in Indonesian includes several types, i.e., pronouns, demonstratives, noun phrases, and named-entities. In this section, we briefly describe the first three types of nominal, while there is no difference of names between Indonesian and English.

Pronouns  Pronouns are a class of words that are used to refer to another noun (Alwi et al., 2010). A personal pronoun is a pronoun which refers to one or more persons. Personal pronouns in Indonesian are differentiated into singular and plural pronouns. Unlike English, pronouns in Indonesian are not

1https://github.com/valentinakania/indocoref

---

*Proceedings of Recent Advances in Natural Language Processing, pages 84–90
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_011*
differentiated by gender, nor by its function in a sentence, i.e., same set of pronouns are used as a subject, an object, or to indicate possession. For example, the word "saya" is a personal pronoun in Indonesian which can be used as a subject pronoun, e.g., "Saya makan" (in English "I am eating."). An object pronoun, e.g., "Pak Bob memanggil saya." (in English "Mr. Bob is calling me."), and a possessive pronoun, e.g., "Ini buku saya." (in English "This is my book.").

Indonesian pronouns can be in form of the clitics (Larasati, 2012). A clitic is a morpheme that is attached to another word or phrase. Clitic pronouns in Indonesian include "-ku" as first-person pronouns, ",-mu" as second-person pronouns, and "-nya" as a third-person pronoun.

**Demonstratives** Demonstrative, also classified as a demonstrative pronoun by Alwi et al. (2010), are words that refer to a noun or noun phrase. In Indonesian, there are two common words that are classified as demonstratives, namely "ini" (in English: "this" or "these") and "itu" (in English: "that" or "those").

**Nouns** Nouns can be seen semantically as words that represent humans, animals, objects, meanings, and concepts that exist in the world as described in Alwi et al. (2010). Syntactically, a noun can be negated with "bukan" (in English: "not to be"), but cannot be negated with "tidak" (in English: "do not"). Nouns can be followed by one or more adjectives, either directly or connected by the word "yang" (in English: "which"). Several Indonesian nouns are multword expression (Suhardijanto et al., 2020), e.g., "kamar tidur" (in English: "bedroom") and "rumah sakit" (in English: "hospital").

**Noun Phrases** A noun phrase may consist of one or more noun(s), pronoun(s), numeral(s), verb(s), adjective(s), and demonstrative(s). A noun phrase in Indonesian is constructed by expanding the noun to the left, with a determiner, or to the right, with modifiers. The initial noun before the expansion is named as the core noun (the head word). While English put modifiers before the head word, modifiers come after the head word in Indonesian, e.g., "buku matematika" vs. "mathematics book".

Here are some rules for expanding a head word into a noun phrase in Indonesian (Alwi et al., 2010).

1. A head word can be expanded to the left with numerals or numeral phrases.

<table>
<thead>
<tr>
<th>satu</th>
<th>meja</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUM</td>
<td>HEAD</td>
</tr>
<tr>
<td>&quot;one table&quot;</td>
<td></td>
</tr>
</tbody>
</table>

2. A head word can be expanded to the right by one or more other nouns (explanatory nouns/EXPN), then followed by a personal pronoun (PP), then it can be followed by a demonstrative word (DEM).

<table>
<thead>
<tr>
<th>meja</th>
<th>kayu</th>
<th>mereka</th>
<th>ini</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEAD</td>
<td>EXPN</td>
<td>PP</td>
<td>DEM</td>
</tr>
<tr>
<td>&quot;this dining table of theirs&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. A head word can be expanded to the right by zero or more adjectives, pronouns or pronominal phrases, followed by a demonstrative.

<table>
<thead>
<tr>
<th>meja</th>
<th>biru</th>
<th>Ibu</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEAD</td>
<td>ADJ</td>
<td>PP</td>
</tr>
<tr>
<td>&quot;mother’s blue table&quot;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**3 Multi-Pass Sieve Coreference Model for Indonesian**

We design multi-pass sieve to resolve coreference problem in Indonesian text. The Multi-Pass Sieve approach works by receiving input in the form of mention pairs \((m_1, m_2)\) and classifying the correlation relationship between the two mentions according to the definition of each sieve layer sequentially and stops when the pair \((m_1, m_2)\) has been declared to have a correlation relationship or when the pair have visited the last sieve. The model is implemented in six tiers, i.e., exact string match, precise constructs, string head match, proper head word match, relaxed head match, and pronoun sieve.

Several other sieves in original Stanford model (Lee et al., 2011) are not adapted in our model due to the differences of linguistic characteristics between English and Indonesian languages. In contrast to English, demonym relation in Indonesian are not expressed by changing words. For example: "Indonesia" and "orang Indonesia" (in English: "Indonesia" and "Indonesian"). A sieve-pass using morphological properties is not included in ours. Indonesian word does not possess gender, number (singular vs. plural), and animacy attribute.

On the other hand, lexical chain sieve is not implemented due to the lack of language resources. Existing Indonesian WordNet (Putra et al., 2008)
does have neither adequate synsets nor lexical semantic relation. In addition, we do not find proper dictionary to construct alias sieve. We also exclude relaxed string match as adverb clauses are out of scope of mention detected in our coreference resolution system. Discourse processing sieve is skipped since the data used in our experiment does not contain any direct sentence.

3.1 Pass 1: Exact String Match
In this first layer of our Multi-Pass Sieve model, each pair of noun phrases is considered coherent if the two strings are the same, regardless of the difference in capitalization and punctuation.

3.2 Pass 2: Precise Constructs
The second layer of the model determines two noun phrases as coreferent if they follow a certain language structure.

**Appositive Relationship**. It is indicated by the position of the two mentions in the sentence in which a mention appears next to another mention. The second mention serves to explain the first mention, which is a proper noun in most cases. Both phrases are usually separated by a punctuation, i.e., comma, semicolon, dash, or brackets.

"[Emma Pillsbury], [salah seorang guru di SMA Ohio], mendengar berita itu."

"[Emma Pillsbury], [one of the teachers at Ohio High School], heard the news."

**Copulative Relationship**. The copulative relationship between two mentions is explanatory, in which both mentions are connected by a copula verb, e.g., "adalah", "merupakan", "yakni", "yaitu", and "ialah", (in English: "is/am/are/was/were"). Like appositive relationship, a mention explains another mention in copulative relationship.

"[Feng Yuxiang] adalah [seorang panglima perang pada masa Republik Tiongkok awal abad-20]."

"[Feng Yuxiang] was [a warlord during the early 20th century Republic of China]."

**Abbreviation Relationship**. The relationship exists when a mention is an abbreviation of another mention. As shown in Budi et al. (2006) and Lee et al. (2011), the abbreviation is detected by using a set of patterns.


"[Pekan Olahraga Nasional] (abbreviated [PON]) is a national sporting event in Indonesia which is held every four years and is attended by all provinces in Indonesia."

In aforementioned example, “PON” is detected as the abbreviation for noun phrase “Pekan Olahraga Nasional”, by matching the first letter of each word in the phrase.

3.3 Pass 3: Strict Head Match
In the third layer of our Multi-Pass Sieve model, two noun phrases are evaluated as coreferent if the head word of both phrases are the same and they also share the same lexical class. There are two variations of the strict head match. (i) **Strict head match**, looks at the similarity of core nouns consisting of one word taken during data preprocessing. (ii) **Full head match**, sees the similarity of the core nouns consisting of several words. The choice of full head in this case is to input words with the POS Tag NOUN or PROPN as the head.

There are three passes that take advantage of the head word or core noun features, i.e., the demonstrative relationship, name abbreviation, and strict head match itself.

**Demonstrative Relationship** The demonstrative relationship between two noun phrases, apart from depending on the position of the phrase in the sentence, also uses demonstrative word classes to determine the relationship. Noun phrases A and B are said to have a demonstrative relationship if one of the phrases contains a demonstrative, and after removing the demonstrative word, the phrase is a sub-phrase of another phrase.


"The Break Up" is [the fourth episode of the fourth season of the musical comedy television series Glee]. [This episode] was screenplayed by Ryan Murphy and directed by Alfonso Gomez-Rejon."
**Short Name** The short name feature sees if one mention is the short name of another mention. The implementation of this feature is as follows: given two noun phrases A and B, where B is shorter than A, B is the short name of A if every word in B is in A, and one of A or B is a proper noun. Examples of cases of short names are found in the mention of people using nicknames or last names in the article text, after the full name is mentioned at the beginning.

### 3.4 Pass 4: Proper Head Word Match

At this layer, the system looks for a specific core noun similarity for PROPN POS Tag. If given two noun phrases A and B which are identified as proper names, the system assesses the two corresponding noun phrases if A and B have core nouns with the same PROPN POS Tag and A and B have compatible attributes. The attribute used in this research is named-entity class.

*Will kemudian memberitahu tunangannya, [guru bimbingan konseling Emma Pillsbury], bahwa ia telah diterima di dewan seni pemerintah. [Emma] tidak mau meninggalkan Lima dalam waktu yang lama.*

“Will then told his fiancée, [counseling teacher Emma Pillsbury] that he had been accepted on the government arts council. [Emma] didn’t want to leave Lima for a long time.”

### 3.5 Pass 5: Relaxed Head Match

In this layer, given two noun phrases A and B, Relaxed Head Match compares whether each word in the head noun A is in the noun phrase B. The head noun A may consist of several words.

*Peternakan Nenek Bebek menjadi [pusat pertemuan keluarga], di mana [pertemuan tersebut] diatur oleh Nenek Bebek.*

“Grandma Duck’s ranch became [the center of the family gathering], where [the meeting] was arranged by Grandma Bebek.”

### 3.6 Pass 6: Pronouns Sieve

The Pronoun layer works anaphorically. If the mention is in the form of pronouns, both words and clitics, the candidate antecedents may only be the noun phrases previously mentioned. In this study, a candidate pronoun will be paired with the closest noun phrase that does not violate the following rules:

- Noun phrases identified as location are eliminated because the pronouns identified by the system are only personal pronouns.
- Noun phrases come before pronouns.
- Especially for pronouns in the form of clitic, the noun phrase attached by the clitic is not considered as a candidate.


“[Princess Stephanie] is the youngest child of Grace Kelly and Rainier III from Monaco. Sometimes [she] becomes a singer, swimsuit designer, model, and circus player.”

### 4 Data Annotation

Since there is no publicly available data of coreference resolution for Indonesian, we construct new data set in this research. We collect the data from Wikipedia in Indonesian language. We filter the Wikipedia pages that fulfill three criteria

1. The pages contain many noun phrases. We hypothesizes they are the Wikipedia pages discussing one of following topics. (i) fictional plots, e.g., subtitles for films, TV show episodes, and novel stories; (ii) biographies (incl. fictional characters); and (iii) historical events or important events.

2. The pages contain significant variation of pronoun and named-entity. We count the number of first, second, third person pronouns, and clitic pronouns in the document by applying string matching. We examine the number of named-entity using the Stanford CoreNLP NER Tagger (Manning et al., 2014) with a model trained from the Indonesian corpus taken from Alfina et al. (2016).

3. The Wikipedia texts have length of 500 to 2000 words.
BCUBED Evaluation Metrics  The BCUBED is a mention-based metric (Bagga and Baldwin, 1998). BCUBED evaluation is done by calculating precision and recall for each mention \( M \), then calculating the final result by weighted-sum of each precision and recall.

\[
\begin{align*}
\text{Precision}_M &= \frac{|M_{\text{result}} \cap M_{\text{gold}}|}{|M_{\text{result}}|} \\
\text{Recall}_M &= \frac{|M_{\text{result}} \cap M_{\text{gold}}|}{|M_{\text{gold}}|} \\
\text{FinalPrecision} &= \sum_{i=1}^{N} w_i \times \text{Precision}_i \\
\text{FinalRecall} &= \sum_{i=1}^{N} w_i \times \text{Recall}_i \\
F1 &= \frac{2 \times \text{FinalPrecision} \times \text{FinalRecall}}{\text{FinalPrecision} + \text{FinalRecall}}
\end{align*}
\]

Weight is generally defined as \( 1/N \), where \( N \) is the number of noun phrases in the document. The F1-score calculation for the BCUBED metric uses a formula that is fundamentally the same as the calculation for the MUC metric as follows:

Table 2 shows that, in general, there is an increasing trend of F1-score on the MUC metric along with the number of passes used. This is due to an increase in recall when one sieve is added one by one. However, when the system gets the best MUC F-measure value of 72.74% if all passes are implemented, the best BCUBED F-measure value is 52.18% when only the first three passes are implemented.

Based on Table 2, coreference resolution has the best precision in the implementation of Pass 1 and 2. This is different from the evaluation results of the Multi-Pass Sieve model in English and the Multi-Pass Sieve concept which depicts that the highest precision is at the top layer. This is due to cases in article text where nominal phrases with the same string refer to different entities. A common case of this allegation is generally the use of the last name to refer to a person, where people with the same surname will be considered the same entity.

Viewed from the group of features per sieve, string similarity feature is a feature that contributes the highest with a recall increase of 21.72% for MUC and 30.03% for BCUBED. The pronoun resolution is also a major contributing layer with recall increases of 33.08% for MUC and 21.99% for BCUBED, despite the decrease in precision and F1 in the BCUBED metric.

The difference in the F-measure trend of the MUC and BCUBED metrics is caused by the diff-
ference in the way the two metrics evaluate two clusters that are merged into one (over-merging). The BCUBED metric evaluates based on mentions, thus penalizing the cases of the merged cluster. The MUC metric evaluates based on mention-links, so that the number of partitions penalized for the two merged clusters is only one link, causing a fairly small penalty.

The decrease in precision in the BCUBED metric is due to the system’s inability to fully detect the antecedents of plural pronouns. Increased precision can be done by increasing the compatibility of attributes between mentions, such as the number and gender attributes which are quite difficult to identify in Indonesian. On the other hand, the head match variation feature, namely proper head match and relaxed head match, does not have much effect on system performance, because the core noun attributes are similar to the previous sieve (strict head match) but cannot take advantage of additional rules such as attribute agreement.

6 Conclusion

We conclude that the Multi-Pass Sieve approach provides a strong baseline performance for coreference resolution task in Indonesian language. The Multi-Pass Sieve approach achieves MUC F-measure up to 72.74% and BCUBED F-measure up to 52.18%. The use of the exact string match feature provides high precision for the Multi-Pass Sieve model, while the increase in recall is also influenced by the Pronoun Sieve. The highest F-measure results for the MUC metric were obtained by Multi-Pass Sieve for the combined implementation of Pass 1-6 and Pass 1-3 models for the BCUBED metric. On the other hand, the proper head match and relaxed head match features do not appear to have a major impact on system performance, contributing to a performance increase of around 1%.

There is still a lot of room to improve the coreference resolution system for Indonesian. In the future, we plan to implement an end-to-end model coreference resolution system, in which the resolved phrases or words can be detected automatically using a more perfect chunking and named-entity recognition system.

Acknowledgement

The authors are supported by Universitas Indonesia through grant “Publikasi Terindeks Internasional (PUTI) Q2: NKB-4060/UN2.RST/HKP.05.00/2020”

References


Solving SCAN Tasks with Data Augmentation and Input Embeddings

Michal Auersperger
Charles University
Faculty of Mathematics and Physics
auersperger@ufal.mff.cuni.cz

Pavel Pecina
Charles University
Faculty of Mathematics and Physics
pecina@ufal.mff.cuni.cz

Abstract

We address the compositionality challenge presented by the SCAN benchmark. Using data augmentation and a modification of the standard seq2seq architecture with attention, we achieve SOTA results on all the relevant tasks from the benchmark, showing the models can generalize to words used in unseen contexts. We propose an extension of the benchmark by a harder task, which cannot be solved by the proposed method.

1 Introduction

Compositionality describes the property of language that the syntactic and semantic aspects of complex language units are composed of the syntactic and semantic aspects of primitive units (Fodor and Lepore, 2002). The SCAN benchmark (Lake and Baroni, 2018) has been designed to assess the ability of current neural networks to utilize compositionality of language to deal with systematic difference between the training and test data distributions (see Table 1 for examples).

We use a novel combination of existing ideas (data augmentation, adding noise and predicting outputs based only on the weighted average of input embeddings) to achieve high accuracy in all the relevant tasks of the SCAN benchmark. An approach working well for all the tasks has not been reported yet.\footnote{The model by Lake (2019) achieves very good results, but it relies on seeing test input sequences (though with modified output) during training.} Seeing the good results, we analyze some underlying assumptions of the tasks and propose a new split of the SCAN data (i.e. a new task) that proves to be more difficult for our approach.

In Section 2, we describe the SCAN benchmark for testing the systematic use of compositionality in detail. Then we present some approaches tested on the benchmark so far (Section 3), introduce our own method of solving the tasks (Section 4) and its results and analysis of some of the decisions (Section 5). In Section 6, we discuss the existing tasks and propose a new one. We conclude the paper in Section 7.

2 The tasks (SCAN dataset)

Lake et al. (2017) discussed the differences between human and machine learning and stressed systematic compositionality as an important ingredient. It makes human learning fast and data efficient, compared to slow and data-hungry training of current deep learning models. Neural networks struggle when provided with familiar concepts in new combinations, while people can use known concepts productively.

The SCAN dataset (Lake and Baroni, 2018) was designed to test this specific aspect of learning. It represents the problem of sequence transduction (sequence to sequence transformation). An agent in a grid world environment is supposed to perform a sequence of primitive actions (output) based on a sequence of commands (input). However, neither the agent nor the environment play any role in the generation of the next examples (there is no ‘state’ as known from reinforcement learning tasks) or in the interpretation of the commands. SCAN is designed to test traditional supervised learning models, namely those that translate from the “command language” into the “action language”.

Some examples taken from the dataset are given in Table 1. There are 13 input tokens (jump, look, run, walk, turn, left, right, and, after, opposite, around, twice, thrice). These are governed by an underlying non-recursive phrase-structure grammar that produces 20,910 possible input sequences in total. Each command sequence is translated into an output action sequence. The output vocabulary contains 6 tokens: (JUMP, LOOK, RUN, WALK,
As far as the training and test data are produced randomly, standard seq2seq models achieve high accuracy of 99.8% (Lake and Baroni, 2018). (The accuracy is computed over whole sequences, i.e. the models make errors in about 0.2% of the testing sequences). Once the training and test data reflect a systematic distribution change, the models struggle. In this work, we address two main sets of tasks presented by SCAN: generalization to a new primitive (JUMP) and generalization to a new combination of learned concepts (AROUND RIGHT).

We discuss them in the following subsections.

### 2.1 Generalization to a new primitive

In this scenario, a primitive command (jump) and its corresponding action are excluded from the training data except for the simplest case where the command and action stand in isolation (jump \(\rightarrow\) JUMP). The test data consist of examples with the primitive in all possible contexts (e.g. jump twice; walk opposite twice after jump around thrice). The models need to learn to generalize the information about the contextualized behavior of other primitives (run twice, walk right) and apply this information to a new primitive (jump twice, jump right).

A simplified version of the task above arises when turn left is held out from the training data instead of jump. The difference is that the corresponding output action (LEFT) still remains in the training data in different contexts: e.g. (jump left \(\rightarrow\) LEFT JUMP)

### 2.2 Generalization to a new combination of learned concepts

Loula et al. (2018) pointed out that while the tasks above are aimed at testing compositionality, there is not much information about the newly added primitives that the models are asked to utilize. We return to this argument later in Section 6. For this reason, a new set of tasks has been added to the benchmark. In these, it is a single combination of well established words that is unique to the test set. There are four such tasks, given here in increasing complexity:

- **jump around right** – unlike the jump task in Section 2.1, here jump appears during training in different contexts (e.g. jump around left; jump right; \(\ldots\)), the only instances left out from the training contain the jump around right sequence
- **Primitive right** – during training, right can only follow the two manner adverbs (around, opposite) but it never follows a verb (jump, look, run, walk, turn) directly
- **Primitive opposite right** – the models are asked to infer the meaning of this sequence (i.e. RIGHT RIGHT ACTION) from seeing examples such as (jump opposite left; look around right, walk right)
- **Primitive around right** – is analogous to the previous task, the only difference being the complexity (length) of the targeted output sequences (jump around right \(\rightarrow\) RIGHT JUMP RIGHT JUMP RIGHT JUMP RIGHT JUMP)

<table>
<thead>
<tr>
<th>Commands</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(look twice) and</td>
<td>(LOOK LOOK) (RIGHT RIGHT)</td>
</tr>
<tr>
<td>(run thrice) after</td>
<td>(LEFT LOOK) (RUN RUN RUN)</td>
</tr>
<tr>
<td>(walk around right)</td>
<td>(RIGHT WALK RIGHT WALK RIGHT WALK RIGHT WALK)</td>
</tr>
<tr>
<td>jump</td>
<td>JUMP</td>
</tr>
<tr>
<td>(walk opposite left) after (run)</td>
<td>(RUN) (LEFT LEFT WALK)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(walk opposite left) after (jump)</td>
<td>(JUMP) (LEFT LEFT WALK)</td>
</tr>
<tr>
<td>jump twice</td>
<td>JUMP JUMP</td>
</tr>
<tr>
<td>(jump thrice) and (walk)</td>
<td>(JUMP JUMP JUMP) (WALK)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 1: Examples from the SCAN dataset (brackets are added for clarity). Each example consists of a command sequence (input) and a corresponding action sequence (output). Two horizontal lines separate the training (top) from the test (bottom) data. This particular split illustrates the jump task, where jump appears in the training data in isolation only.
3 Existing methods

We describe notable approaches to the SCAN benchmark, selected because of their good performance and/or relevance to our proposed method:

**Seq2seq** model with the LSTM encoder and decoder and the attention (Bahdanau et al., 2015) mechanism is the baseline method evaluated by Lake and Baroni (2018).

Dessi and Baroni (2019) employ convolutional neural networks (CNN) and observe substantial increase in accuracy in the SCAN tasks. This seems to support the conclusion of Bastings et al. (2018), who argue that SCAN tasks inherently prefer simpler models (mainly because of very limited temporal dependencies within the output sequences).

Andreas (2020) employs the baseline recurrent models (Lake and Baroni, 2018) together with a general data augmentation technique (GECA) to expand the training data. After the augmentation, the training set contains 5% (JUMP) and 1% (AROUND RIGHT) of the test data, which leads to an increase in performance.

Russin et al. (2020) modify the baseline model by using different attention values: in each decoding step, they keep the computation of the attention weights over the input sequence, but for attention values, the encoder hidden states are replaced with a second set of input word embeddings. These word embeddings are different from the embeddings used as the encoder input. It is an attempt to separate syntactic (computation of attention weights) and semantic (attention values) information (SyntAttn).

Li et al. (2019), in the same vein, separate the flow of information into two streams, which they call primitive (i.e. semantic) and functional (i.e. syntactic). Again, syntactic embeddings are used to determine the attention weights over the input sequence and semantic embeddings are used as the actual values the attention mechanism produces. Moreover, the approach uses regularization (adding noise to the embeddings, $L_2$ norm) leading to more stable training (and better results) compared to the previous method (Li19).

Finally, Lake (2019) uses a seq2seq model with the combination of external memory, data augmentation and meta-learning and achieves good performance on multiple SCAN tasks (MetaSeq2seq).

4 Model

Our architecture is a straightforward extension of the baseline LSTM seq2seq model with attention (Lake and Baroni, 2018). We modify the attention mechanism by using input word embeddings as the attention values (as opposed to using contextualized representations produced by the encoder). This can also be seen as a simplified version of the architectures introduced by Li et al. (2019); Russin et al. (2020). Unlike them, we train only one set of embeddings that are used both as the encoder input and as attention values. Also, we use the traditional autoregressive decoder (the previous predicted output serves as the input for the decoder at each step), which is not the case in the two cited approaches.

We added standard Gaussian noise to the input embeddings during training, as suggested by Li et al. (2019), but did not use the $L_2$ regularization.

We also experimented with weight tying, known for example, from some language modelling literature (Inan et al., 2017). We tried to avoid using a separate output layer and instead, we computed similarity (dot product) of the attention output with the output token embeddings (the embeddings used by the decoder) and used these vectors as logits.

Similarly to Lake (2019), we experiment with adding artificial primitives to the training data. We hope this would make the encoder more robust as it should recognize the syntactic patterns (X opposite right) rather than memorize each instance (walk opposite right) using the available embedding (walk).

For the jump task, we introduce artificial action commands action1, action2, action3, . . . with the corresponding actions ACTION1, ACTION2, ACTION3, . . . For all the other tasks (turn, jump around right, Primitive right, Primitive opposite right, Primitive around right), we introduce artificial directions: dir1 → DIR1, dir2 → DIR2, dir3 → DIR3 . . . Unlike Lake (2019), we are not permuting the command – action assignment during training and we do not introduce the held-out phrases to either the input or the output of the training examples.

4.1 Training details

We trained all the models using the Adam optimizer with the learning rate 0.001 exponentially decaying to 0.0001 over 60 epochs (regardless of the task). The batch size was set to 32. We experimented with different sizes of the input and output embeddings and LSTM dimensions (64, 128, 256, 512), and the
Method & jump & around right &  
seq2seq & 1.2 & 2.46 ± 2.68 &  
CNN & 69.2 ± 8.2 & 56.7 ±10.2 &  
GECA & 87 ± 2 & 82 ± 4 &  
SyntAttn & 91.0 ±27.4 & 28.9 ± 34.8 &  
Li19 & 98.8 ± 1.4 & 83.2 ± 13.2 &  
MetaSeq2seq & 99.95 ± 0.08 & 99.96 ± 0.08 &  
ours & 98.90 ± 1.40 & 99.82 ± 0.47 &  

Table 2: Test accuracy (mean ± std %) on the two most challenging SCAN tasks. Results are given as reported by their authors. Seq2seq is from Lake and Baroni (2018) for the jump and Loula et al. (2018) for the around right task, CNN from Dessi and Baroni (2019), GECA from Andreas (2020), SyntAttn from Russin et al. (2020), Li19 from Li et al. (2019) and MetaSeq2seq from Lake (2019). Results are reported over 5 random seeds with the exception of GECA (10 seeds), SyntAttn (median value, 25 seeds) and ours (25 seeds).

encoder layers (1, 2). The decoder was initialized by the final hidden state of the encoder. We did not employ dropout and we clipped gradients whose norm was larger than 1. Teacher forcing was used during training. In preliminary experiments, we settled on the model with one layer bidirectional LSTM encoder, embedding size of 256 and LSTM dimension of 256. We experimented with adding 10, 50, 75 and 100 artificial command – action pairs.

5 Results and analysis

We provide the test accuracy of our model on the two most widely discussed SCAN tasks (jump and around right) in Table 2. For completeness, the results for the remaining tasks are given in Table 3. The reported model was trained using data augmentation with 75 additional command – action pairs, Gaussian noise added to the input embeddings during training and no parameter tying. Below, we investigate the effect of these choices. If not stated otherwise, we always report a mean and standard deviation computed over 25 runs with different random seeds.

5.1 Data augmentation

Arguably, the biggest benefit comes from the data augmentation. We illustrate this in Table 4, where we compare the baseline model trained with and without data augmentation on all the discussed SCAN tasks. With the exception of the jump and around right tasks, data augmentation seems to be enough to achieve good results.

5.2 Parameter tying

The effect of replacing a separate output layer with a dot product with decoder embeddings was most prominent with less intensive data augmentation. When adding only 10 additional command – action pairs to the training data of the around right task, our top model achieved the mean accuracy of 59.47% (±18.27%) with the output layer and 90.4% (±7.5%) without it. The effect disappeared with heavier data augmentation and/or with other tasks. Therefore, parameter tying was not included in the overall results.

5.3 Input embeddings as attention values

In contrast, the other architectural change with respect to the seq2seq baseline, i.e. using the input embeddings as attention values, had an effect even with stronger data augmentation. Models with modified attention trained with 100 additional command – action pairs achieved the mean accuracy of 99.44% (±0.61%) on the jump task, compared to 68.51% (±10.67%) achieved by corresponding models, where the attention values were formed by the encoder output.

5.4 Embedding noise

Adding standard Gaussian noise to the input embeddings plays also an important role by making the training more stable. Our overall best model without the noise added in training achieves only 89.12% (±12.91%) accuracy on the jump task (compared to 98.9% (±1.4%) by the same model with the noise).

5.5 Single jump proportion

During training on the jump task, the models were sensitive to the proportion of isolated jump examples in the training data. In the original dataset, such examples are upsampled so that they form about 10% of the training data (without this, models would encounter jump only once each episode). We found the best results, when these examples were treated as all the other ones, i.e. they were also eligible for command replacement with uniform probability.

6 Discussion

We show that task-specific data augmentation with simple architecture modification leads to good re-
Method | turn left | jump ar. right | right | opposite right
--- | --- | --- | --- | ---
seq2seq | 90.3 | 98.43 ± 0.54 | 23.49 ± 8.09 | 47.62 ± 17.72
SyntAttn (Russin et al., 2020) | 99.9 ± 0.16 | 98.9 ± 2.3 | 99.1 ± 1.8 | 10.5 ± 8.8
Li et al. (2019) | 99.7 ± 0.4 | 100.0 ± 0.0 | 99.7 ± 0.5 | 89.3 ± 5.5
ours | 99.99 ± 0.02 | 99.98 ± 0.05 | 99.99 ± 0.04 | 99.98 ± 0.04

Table 3: Test accuracy (mean ± std %) on the remaining SCAN tasks. The baseline (seq2seq) results were reported by Lake and Baroni (2018) for the turn left task and by Loula et al. (2018) for the other tasks. Results are reported over 5 random seeds with the exception of ours (25 seeds).

<table>
<thead>
<tr>
<th>Task</th>
<th>extra0</th>
<th>extra100</th>
</tr>
</thead>
<tbody>
<tr>
<td>jump</td>
<td>0.11 ± 0.11</td>
<td>57.92 ± 15.91</td>
</tr>
<tr>
<td>around r.</td>
<td>0.0 ± 0.0</td>
<td>98.69 ± 1.33</td>
</tr>
<tr>
<td>turn l.</td>
<td>49.55 ± 8.89</td>
<td>99.92 ± 0.24</td>
</tr>
<tr>
<td>jump ar. r.</td>
<td>82.97 ± 11.14</td>
<td>99.62 ± 0.96</td>
</tr>
<tr>
<td>right</td>
<td>3.18 ± 2.3</td>
<td>99.90 ± 0.2</td>
</tr>
<tr>
<td>opposite r</td>
<td>0.04 ± 0.07</td>
<td>99.97 ± 0.04</td>
</tr>
</tbody>
</table>

Table 4: The effect of data augmentation: the same baseline seq2seq model trained on the original data (extra0) and with 100 additional command–action pairs (extra100). Test accuracy (mean ± std %, 25 seeds)

We hypothesize that current methods (including ours) manage to solve this compositionality challenge without using much compositionality. Specifically, what is enough for all the tasks as they have been defined is to learn to attend to correct positions and then use a one to one mapping from the input to the output space. This “align and translate” approach (Bahdanau et al., 2015) approach exhibits some degree of compositionality. However, having learned somewhat complex information (say in the case of grammatical words such as around or opposite), this needs to be combined with just the identity of another word. This was actually the motivation of Loula et al. (2018) to introduce new tasks into the benchmark (e.g. around right, see Section 2.2). We believe that in this respect, the new tasks suffer from the same weaknesses as the old ones (as evidenced by the possibility of easily using data augmentation).

### 6.1 Proposed task

Based on the presented results and the discussion above, we propose another task for the SCAN benchmark: around Direction twice. ²

Here, we form the test set by selecting all the SCAN examples where the input string contains the sequence around right/left twice. Ideally, a model should be able to combine the information about doing something while turning around and doing something twice based on such examples as jump around left, jump around right thrice, jump twice, jump opposite right twice.

At face value, this task is almost identical to the tasks discussed in Section 2.2 (e.g. around right): we held out a certain combination of well established words from the training and tested on them. What we believe is the substantial difference is the fact that neither around nor twice have a direct ‘translation’ counterpart in the output ‘language’. This makes the proposed task less suitable for the above-mentioned “align and translate” approach.

Another practical effect of this is that it is not very obvious how data augmentation could be used. We would need to generate alternatives to either twice, which would quickly lead to uncomfortably long output sequences (e.g. run around right thirty-
seven times) and/or around, which seems even more obscure.

We trained 3 types of models for the new task: the baseline seq2seq architecture without modifications; our seq2seq model with added noise and modified attention values; the same model with 100 additional augmented directions (e.g. run around dir3 thrice). All the models failed with results: 10.05%(±2.30%), 8.78%(±2.48%) and 5.53%(±2.12%) respectively (each model was trained with 5 different random seeds).

7 Conclusion

The SCAN benchmark is a very accessible dataset used for investigating compositionality. Cherry-picking from previous research, we introduced the first approach that is successful in all the relevant tasks from the benchmark. Analyzing the important features defining our approach, we introduced a new task, where this approach fails. We hypothesize the new task is more challenging than the existing ones since it tests for deeper aspects of compositionality.

Acknowledgments

The work was supported by the grant 978119 of the Grant Agency of Charles University in Prague (GAUK) and by the grant 19-26934X of the Czech National Science Foundation (GACR).

References


PyEuroVoc: A Tool for Multilingual Legal Document Classification with EuroVoc Descriptors

Andrei-Marius Avram, Vasile Păis, Dan Tufis
Research Institute for Artificial Intelligence, Romanian Academy
Bucharest, Romania
{andrei.avram,vasile,tufis}@racai.ro

Abstract

EuroVoc is a multilingual thesaurus that was built for organizing the legislative documentary of the European Union institutions. It contains thousands of categories at different levels of specificity and its descriptors are targeted by legal texts in almost thirty languages. In this work we propose a unified framework for EuroVoc classification on 22 languages by fine-tuning modern Transformer-based pretrained language models. We study extensively the performance of our trained models and show that they significantly improve the results obtained by a similar tool - JEX - on the same dataset. The code and the fine-tuned models were open sourced, together with a programmatic interface that eases the process of loading the weights of a trained model and of classifying a new document.

1 Introduction

EuroVoc\textsuperscript{1} is a multilingual thesaurus which was originally built up specifically for processing the documentary information of the EU institutions. The covered fields are encompassing both European Union and national points of view, with a certain emphasis on parliamentary activities. The current release 4.4 of EuroVoc was published in December 2012 and includes 6,883 IDs for thesaurus concepts (corresponding to the preferred terms), classified into 21 domains (top-level domains), further refined into 127 subdomains. Additional forms of the preferred terms are also available and are assigned the same ID, subdomains and top-level domains.

Multilingual EuroVoc thesaurus descriptors are used by a large number of European Parliaments and Documentation Centres to index their large document collections. The assigned descriptors are then used to search and retrieve documents in the collection and to summarise the document contents for the users. As EuroVoc descriptors exist in one-to-one translations in almost thirty languages, they can be displayed in a language other than the text language and give users cross-lingual access to the information contained in each document.

One of the most successful recent approaches in document and text classification involves fine-tuning large pretrained language models on a specific task (Adhikari et al., 2019a; Nikolov and Radevchev, 2019). Thus, in this work we propose a tool for classifying legal documents with EuroVoc descriptors that uses various flavours of Bidirectional Encoder from Transformers (BERT) (Devlin et al., 2019), specific to each language. We evaluated the performance of our models for each individual language and show that our models obtain a significant improvement over a similar tool - JEX (Steinberger et al., 2012). The The models were further integrated into the RELATE platform (Păis et al., 2020) and an API was provided through the PythonPackage Index (PyPi) interface\textsuperscript{2} that facilitates the classification of new documents. The code used to train and evaluate the models was also open-sourced\textsuperscript{3}.

The rest of the paper is structured as follows. Section 2 presents other works in the direction of EuroVoc classification. Section 3 provides several statistics with regard to the corpus used to train and test the models in the tool. Section 4 presents the approach used in fine-tuning the pretrained language models and the exact BERT variants used for each language, together with a vocabulary statistics of the model’s tokenizer on the legal dataset. Section 5 outlines our evaluation setup and the results of our experiments, while Section 6 presents

\textsuperscript{1}https://data.europa.eu/data/datasets/eurovoc
\textsuperscript{2}https://pypi.org/project/pyeurovoc/
\textsuperscript{3}https://github.com/racai-ai/pyeurovoc
the programmatic interface. Finally, the paper is concluded in Section 7.

2 Related Work

JEX (Steinberger et al., 2012) is a multi-label classification software developed by Joint Research Centre (JRC), that was trained to assign EuroVoc descriptors to documents that cover the activities of the EU. It was written entirely in Java and it comes with 4 scripts (both batch and bash) that allows a user to pre-process a set of documents, train a model, postprocess the results and evaluate a model. Each script is easily configurable from a properties file that contains most of the necessary parameters. The toolkit also comes with a graphical interface that allows a user to easily label a set of new documents (in plain text, XML or HTML) or to train a classifier on their own document collections.

The algorithm used for classification was described in (Pouliquen et al., 2006) and it consists in producing a list of lemma frequencies from normalized text, and their weights, that are statistically related to each descriptor, entitled in the paper as associates or as topic signatures. Then, to classify a new document, the algorithm picks the descriptors of the associates that are the most similar to the list of lemma frequencies of the new document. The initial release consisted of 22 pretrained classifiers, each corresponding to an official EU language.

Boella et al. (Boella et al., 2012), while focusing on the Italian JRC-Acquis-IT corpus, presents a technique for transforming multi-label data into mono-label that is able to maintain all the information as in (Tsoumakas and Katakis, 2007), allowing the use of approaches like Support Vector Machines (SVM) (Joachims, 1998) for classification. Their proposed method allows an F1 score of 58.32 (an increase of almost 8% compared to the JEX score of 50.61 for the Italian language).

Šarić et al. (Šarić et al., 2014) further explores SVM approaches for classification of Croatian legal documents and report an F1 score of 68.6. Unfortunately, this is not directly comparable with the JEX reported results since the training corpus is a different collection called NN13205. Furthermore, the categories being used for the gold annotation represent an extended version of EuroVoc for Croatian, called CroVoc.

Studies, such as in (Collobert et al., 2011), have shown that neural word embeddings can store abundant semantic meanings and capture multi-aspect relations into a real-valued matrix, when trained on large unlabeled corpora using neural networks. Considering a vocabulary $V$, an embeddings representation can be learned by means of a neural network resulting into an association of a real-valued vector $W_n$ of size $n$ to each word. Two neural network methods for automatically learning distributed representation of words from a large text corpus can be considered: Skip-gram and continuous bag of words (CBOW) (Mikolov et al., 2013). In the case of CBOW, a neural network is trained to predict the middle word given a context, while Skip-gram uses a single word as input and tries to predict past and future words. Bojanowski et al. (Bojanowski et al., 2017) introduced a method for runtime representation for unknown words by means of averaging pre-trained character n-grams, also known as subword information.

BERT has also been used to classify legal documents with EuroVoc labels, with most of the work focusing on the English language. In (Chalkidis et al., 2019), the authors studied the problem of Large-Scale Multi-Label Text Classification (LMTC) for few- and zero-shot learning and released a new dataset composed of 57k samples from EUROLEX on which several models were tested. The results showed that BERT obtained superior performance in all but zero-shot classification.

3 Dataset Statistics

The training of BERT models for the 22 languages was done using the same dataset that was used for training the JEX models. The dataset is composed of two parallel corpora from the legal domain, JRC-Acquis (Steinberger et al., 2006) and the Publications Office of the European Union (OPOCE), that were manually labeled with over 6,700 EuroVoc descriptors identifiers (ID). The EuroVoc descriptors are hierarchically organised and can be converted into higher level Microthesaurus labels (MT) and further into top-level domains (DO).

The number of documents in the dataset range from 17,858 documents in Maltese to 41,989 documents in French. Each document is labeled with multiple ID descriptors, having an average of 6 ID descriptors which can be equivalently converted to 5 MT descriptors or 4 DO descriptors. In Figure 1 we depict the distribution of the average number of ID, MT and DO descriptors per document, together with the difference between the minimum and the
maximum number of documents per descriptor.

The ID, MT and DO descriptors distributions are also highly unbalanced. Figure 2 depicts the number of documents of the most frequent ID, MT and DO descriptors, organised in groups of 50, 5 and 1, respectively. Each group contains the sum of the number of documents that are labeled with each descriptor in the respective group. As it can be observed, in each subplot, the number of documents that contain the descriptors from the first few groups is higher than the number of document that contain all the other descriptors.

4 Methodology

The proposed approach for classifying the legal documents found in the two corpora is to fine-tune a pre-trained BERT on each of the 22 languages. We follow the method introduced in (Devlin et al., 2019) where a simple feed-forward network with the weights $W \in \mathbb{R}^{E \times M}$, $E$ is embedding size of BERT and $M$ is the number of classes, and bias $b \in \mathbb{R}^M$ is put on top of the embedding of the first token $C'(\text{[CLS]})$ to create the output logits of the classification problem for the ID descriptors$^4$.

The sigmoid $\sigma$ function is then applied to produce independent probability distributions $\hat{y}$ over each class:

$$\hat{y} = P(y|x) = \sigma(C^T W + b)$$  \hspace{1cm} (1)

Additionally, a dropout of 0.1 is applied on the feed-forward layer to regularize the model.

The models are optimized by reducing the loss $\mathcal{L}$ computed as the average binary cross-entropy between the output probabilities $\hat{y}$ and the target classes $y$, over the $M$ classes (ID descriptors):.

$$\mathcal{L} = -\frac{1}{M} \sum_{i=1}^{M} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$ \hspace{1cm} (2)

Because the flavours of BERTs vary from one language to another, the choice of the initial models for each language was made by using the following heuristic, based on the corpora used for pretraining: Legal > Monolingual (Mono) > Wikipedia (Wiki) > Multilingual (Multi). The heuristic is experimentally supported by (Chalkidis et al., 2020) that showed that language models obtain superior performance on the legal domain when they are pretrained on legal corpora and by (Pyysalo et al., 2020) that outlined the superiority of BERTs pretrained on monolingual Wikipedia over multilingual BERT (mBERT). Also, it was empirically proven that the performance of the language models improves when they are pretrained on larger

---

$^4$The MT and DO descriptors are predicted by using a direct mapping scheme
corpora (Liu et al., 2019) and for this reason we expect most of the general monolingual models to obtain better result than Wikipedia BERTs. Thus, given the existing open-sourced models for each language, we use the following taxonomy in our experiments: (Figure 3) 5:

- **Legal**: English (en) - Legal BERT (Chalkidis et al., 2020).

- **Mono**: Danish (da) - Danish BERT 6, German (de) - German BERT 7, Greek (el) - Greek BERT (Koutsikakis et al., 2020), Spanish (es) - Spanish BERT (Canete et al., 2020), Estonian (et) - EstBERT (Tanvir et al., 2020), Finnish (fi) - Finnish BERT (Virtanen et al., 2019), French (fr) - CamemBERT (Martin et al., 2020), Hungarian (hu) - huBERT (Erzsébet et al.), Italian (it) - Italian BERT 8, Dutch - BERTje (de Vries et al., 2019), Polish (pl) - PolBERT (Kleczer, 2020), Portuguese (pt) - BERTimbau (Souza et al., 2020), Romanian (ro) - Romanian BERT (Dumitrescu et al., 2020), Swedish (sv) - Swedish BERT (Malmsten et al., 2020).

- **Wiki**: Bulgarian (bg) - WikiBERT-BG, Czech (cs) - WikiBERT-CS, Lithuanian (lt) - WikiBERT-LT, Latvian (lv) - WikiBERT-LV, Slovak (sk) - WikiBERT-SK, Slovene (sl) - WikiBERT-SL.

- **Multi**: Maltese (mt) 9.

The vocabulary of BERT plays an important role in the final performance of the model. Broadly speaking, the fewer tokens each word is split into, the better the language model is expected to perform. In Table 1 we depict the average number of tokens per word and the average number of unknown (UNK) tokens per word on the legal dataset for each tokenizer of the 22 BERT models. As it can be observed, the lowest number of tokens per word is achieved by the Spanish BERT with 1.25 followed closely by the Legal BERT with 1.28. When looking at unknown tokens per word, CamemBERT tokenizer leads the leaderboard with no unknown words when tokenizing the dataset. On the other hand, the highest number of tokens and unknown tokens per word was achieved on Maltese due to use of mBERT instead of a monolingual

---

5To the best of our knowledge, not all languages have publications for their monolingual versions of BERT, so we attached a corresponding URL in these cases.

6https://github.com/botxo/nordic_bert

7https://huggingface.co/bert-base-german-cased

8https://huggingface.co/dbmdz/bert-base-italian-cased

9https://huggingface.co/bert-base-multilingual-cased
model.

The legal documents in the corpus can be rather long and exceed the maximum number of tokens of 512 allowed by the BERT models. To mitigate this, we simply keep only the first 512 in the document and discard the rest. This method has been shown to lead to approximately the same performance as considering the whole document (Chalkidis et al., 2019).

5 Experiments

5.1 Evaluation Setup

Because the original splits used for training and evaluating the JEX models were not made publicly available, we united the JRC-Acquis and OPOCE datasets for each language and split it 5 times in train, validation and test sets using different seeds. Moreover, in order to preserve the class balance across the sets in one split, we employed an iterative stratification splitting approach as proposed in (Sechidis et al., 2011) and kept an approximate ratio of 80% train, 10% validation and 10% test for fine-tuning and evaluating the pre-trained language models and a ratio of 90% train and 10% test for training and evaluating the JEX models.

The pre-trained language models were fine-tuned for 30 epochs, using a batch size of 8 and the AdamW optimizer (Loshchilov and Hutter, 2018) whose learning rate was decayed by a linear scheduler peaking at 6e-5, in order to reduce the oscillations in the later stages of training due to the high values of the learning rate. We also clipped the gradients (Pascual et al., 2013) whose norm had a value over 5 and used a learning rate warm-up over the first epoch to alleviate the effects of forgetting the knowledge learned by Transformer models in the pre-training phase. The final weights of each fine-tuned language model were the ones that obtained the lowest loss on the validation set during training.

The training and evaluation of JEX models followed the approach described in (Steinberger et al., 2012). Both JEX models and the pre-trained language models were trained five times on each split with the results averaged over all test splits. We also used the validation splits for early stopping and fine-tune the hyperparameters of the BERT models.

5.2 Evaluation Metrics

Most used metrics for evaluating LMTC models are the precision ($P@K$), the recall ($R@K$) and their harmonic mean, known as F1 score ($F1@K$), over the top $K$ predicted labels. These metrics usually unfairly penalize documents that have fewer or more labels than $K$, but we still use them because they allow a direct comparison with the original results of JEX. The three metrics are defined as follows:

$$P@k = \frac{1}{k} \sum_{l \in r_k(\hat{y})} y_l$$

$$R@k = \frac{1}{n} \sum_{l \in r_k(y)} y_l$$

$$F1@k = 2 \cdot \frac{P@k \cdot R@k}{P@k + R@k}$$

where $k$ is the number of labels to be used for comparison, $n$ is the number of true labels of the respective document, $y \in \{0, 1\}^L$ is the vector of the true labels, $\hat{y} \in \mathbb{R}_L$ is the vector of predicted labels and $r_k(\hat{y})$ is a function that selects the index of the $k$th largest value in the prediction labels.

As the statistics in Section 3 have shown, the average number per document, of ID descriptors is 6, of MT descriptors is 5 and of DO descriptors is 4. Thus, we evaluate both JEX and BERT by using the F1 score for 6 labels on ID descriptors ($F1@6$), for 5 labels on MT descriptors ($F1@5$) and for 4 labels on DO descriptors ($F1@4$).

5.3 Results

The results for both JEX and the BERT models on the 22 languages by using the cross-validated dataset are outlined in Table 2. The BERT models obtained a significant improvement over JEX on each language, ranging from an enhancement of 21.54% (el), 14.85% (fr) and 9.65% (el) to an enhancement of 37.06% (sk), 25.94% (ro) and 9.65% (el) to an average number per document of 5 for ID, 63.15% for MT and 21.54% for DO descriptors, respectively. The highest F1 scores with JEX were achieved on German with 50.65% $F1@6$, 63.15% $F1@5$, 72.23% $F1@4$, and the highest F1 scores with the BERT models were achieved on Slovenian with 84.90% $F1@6$, 87.37% $F1@5$, 91.72% $F1@4$ for ID, MT and DO descriptors, respectively. On the other hand, the lowest scores were obtained on Maltese. This might be due to the low number of
Table 2: Evaluation results of JEX and BERT for ID, MT and DO descriptors.

<table>
<thead>
<tr>
<th>Language</th>
<th>JEX</th>
<th>BERT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ID</td>
<td>MT</td>
</tr>
<tr>
<td></td>
<td>F1@6</td>
<td>F1@5</td>
</tr>
<tr>
<td>Bulgarian (bg)</td>
<td>47.66</td>
<td>60.45</td>
</tr>
<tr>
<td>Czech (cs)</td>
<td>47.81</td>
<td>60.03</td>
</tr>
<tr>
<td>Danish (da)</td>
<td>49.79</td>
<td>62.45</td>
</tr>
<tr>
<td>German (de)</td>
<td>50.65</td>
<td>63.15</td>
</tr>
<tr>
<td>Greek (el)</td>
<td>49.65</td>
<td>61.34</td>
</tr>
<tr>
<td>English (en)</td>
<td>50.15</td>
<td>62.86</td>
</tr>
<tr>
<td>Spanish (es)</td>
<td>50.42</td>
<td>62.47</td>
</tr>
<tr>
<td>Estonian (et)</td>
<td>48.98</td>
<td>61.19</td>
</tr>
<tr>
<td>Finnish (fi)</td>
<td>49.41</td>
<td>61.65</td>
</tr>
<tr>
<td>French (fr)</td>
<td>50.61</td>
<td>62.87</td>
</tr>
<tr>
<td>Hungarian (hu)</td>
<td>49.71</td>
<td>62.22</td>
</tr>
<tr>
<td>Italian (it)</td>
<td>49.53</td>
<td>61.15</td>
</tr>
<tr>
<td>Lithuanian (lt)</td>
<td>48.26</td>
<td>61.20</td>
</tr>
<tr>
<td>Latvian (lv)</td>
<td>47.79</td>
<td>59.14</td>
</tr>
<tr>
<td>Maltese (mt)</td>
<td>44.36</td>
<td>55.99</td>
</tr>
<tr>
<td>Dutch (nl)</td>
<td>50.64</td>
<td>62.40</td>
</tr>
<tr>
<td>Polish (pl)</td>
<td>46.86</td>
<td>59.15</td>
</tr>
<tr>
<td>Portuguese (pt)</td>
<td>50.41</td>
<td>62.55</td>
</tr>
<tr>
<td>Romanian (ro)</td>
<td>47.13</td>
<td>60.18</td>
</tr>
<tr>
<td>Slovak (sk)</td>
<td>46.34</td>
<td>58.36</td>
</tr>
<tr>
<td>Slovenian (sl)</td>
<td>49.96</td>
<td>62.63</td>
</tr>
<tr>
<td>Swedish (sv)</td>
<td>50.32</td>
<td>62.21</td>
</tr>
</tbody>
</table>

documents compared to the other languages (Steinberger et al., 2012), but also because, in the case of the BERT variant, we use a multilingual model instead of a monolingual one.

Figure 4 depicts the $F1@6$-scores obtained by the BERT models on multi-label ID classification relative to the scores obtained by JEX models in the same language. One interesting aspect that can be observed in the plot is that although the mBERT used for Maltese obtained the lowest $F1$-score, its relative improvement over JEX is higher than of the other three languages that use monolingual models: Greek, Hungarian and French, mostly because the $F1@6$-score obtained by JEX on Maltese is lower when compared to the other three and thus the difference would normally be larger.

The variance of scores between languages is higher for the BERT models than for the JEX models. This happens because the BERT models were pretrained beforehand on various corpora taken from different sources and aspects like the quality of the corpus or the domain match greatly influenced the resulted fine-tuning performance. One interesting result is that WikiBERT obtained some of the highest scores and that Legal BERT also did not perform as well as expected, thus partially contradicting the heuristic introduced in the previous section. Due to time and resource constraints, we leave a detailed study of the heuristic for future work.

5.4 Comparison with State-of-the-Art

The state-of-the-art (SOTA) for EuroVoc multi-label classification was presented in (Chalkidis et al., 2019) by using the original BERT-base. The model was trained and evaluated on EUR-LEX, an English corpus introduced in the same paper. We evaluated our English model (Legal BERT), trained on JRC-Acquis and OPOCE, and report in Table 3 the R-Precision ($RP$), the normalized discounted cumulative gain (nDCG) and the F1-micro score for extracting 5 ID descriptors on the test set of EUR-LEX. Our model obtained a $RP@5$ of 81.2%,
Figure 4: Performance of BERT models on ID multi-label classification relative to the performance of JEX models on the same language.

<table>
<thead>
<tr>
<th>Model</th>
<th>RP@5</th>
<th>nDCG@5</th>
<th>Micro-F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN-LWAN</td>
<td>71.6</td>
<td>74.6</td>
<td>64.2</td>
</tr>
<tr>
<td>BIGRU-LWAN</td>
<td>76.6</td>
<td>79.6</td>
<td>69.8</td>
</tr>
<tr>
<td>BIGRU-LWAN-L2V</td>
<td>77.5</td>
<td>80.4</td>
<td>71.1</td>
</tr>
<tr>
<td>BERT-base</td>
<td>79.6</td>
<td>82.3</td>
<td>73.2</td>
</tr>
<tr>
<td>Legal BERT (ours)</td>
<td>81.2</td>
<td>83.4</td>
<td>79.6</td>
</tr>
</tbody>
</table>

Table 3: Our Legal BERT model compared with BERT-base and BIGRU-LWAN on EUR-LEX test.

Other extensive document classification experiments with BERT were conducted by (Adhikari et al., 2019a), without specific consideration to EuroVoc labels. They used a similar approach to ours by introducing a fully-connected layer over the embedding of the first token \([CLS]\). Furthermore, the paper also presents the results for a knowledge distillation process (Hinton et al., 2015) from the fine-tuned BERT-large into the previous SOTA (Adhikari et al., 2019b), a much smaller network, obtaining better results than BERT-base on the evaluated datasets, but still behind BERT-large.

5.5 Response Time

The response time of the API was tested on a CPU - Intel Xeon Silver 4210 - and on a GPU - Nvidia Quadro RTX 5000. Because the pretrained language models have mostly the same dimension, we made an inference time analysis only for the English language on CPU and GPU.

6 Programmatic Interface

To ease the loading of models and the classification of documents, we created a programmatic interface in Python that can be installed using PyPi with the command `pip install pyeurovoc`. Once the library is installed, a BERT model is simply created by instantiating the class `EuroVocBERT` with one of the 22 languages. The class will either
download the fine-tuned model from the repository or will use a local cached version of it. Finally, the classification of a document is made by calling the instantiated model with the document text.

More detailed information about the API and how custom pre-trained BERT models can be fine-tuned on the dataset can be found at the source repository. An example of API usage is presented in Appendix A.

7 Conclusion and Future Work

Document classification remains a relevant problem in nowadays society, aiding companies and government institutions to index their large textual database. This paper presented a tool for classifying legal documents with EuroVoc descriptors that use various Transformer-based language models, fine-tuned on the 22 languages that are found in JRC-Acquis and OPOCE. We thoroughly evaluated the models on multiple splits of the data and the results showed that they significantly improve the performance obtained by another similar tool - JEX. The pretrained models were made publicly available and they can be easily used to classify new documents using our API.

One direction for possible future work is to improve the inference speed of the models by either distilling their knowledge in a smaller network (Hinton et al., 2015) or quantizing their weights (Yang et al., 2019). Furthermore, we intend to include our results for legal document classification in language specific NLP benchmarks such as KLEJ for Polish (Rybak et al., 2020), LiRo for Romanian (Dumitrescu et al., 2021) or EVALITA4ELG for Italian (Patti et al., 2020).

Acknowledgments

This research was supported by the EC grant no. INEA/CEF/ICT/A2017/1565710 for the Action no. 2017-EU-IA-0136 entitled “Multilingual Resources for CEF.AT in the legal domain” (MAR-CELL).

References


Frane Sari9, Bojana Dalbelo Basic, Marie-Francine Moens, and Jan Snajder. 2014. Multi-label classification of croatian legal documents using euvoc thesaurus. In Proceedings of SPLeT-Semantic processing of legal texts: Legal resources and access to law workshop. ELRA.


Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Toma9 Erjavec, Dan Tufis, and Daniel Varga. 2006. The jrc-acquis: A multilingual
aligned parallel corpus with 20+ languages. In Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC’06).


Appendix A. API Code Snippet

The following is a code snippet that loads the BERT model for English from the checkpoint repository and classifies a document, given its text.

```python
from pyeurovoc import *

model = EuroVocBERT(lang="en")
outputs = model(<document text >)
```

The code snippet will return a dictionary of ID descriptors and confidence scores. The number of labels returned by the model for ID descriptors type is controlled by the num_labels.

```python
{
  <ID_label1>: <score1>,
  <ID_label2>: <score2>,
  ...
}
```
TEASER: Towards Efficient Aspect-based SEntiment analysis and Recognition

Vaibhav Bajaj, Kartikey Pant, Ishan S Upadhyay, Srinath Nair and Radhika Mamidi
IIIT Hyderabad
Hyderabad, India
{vaibhav.bajaj,kartikey.pant}@research.iiit.ac.in
{ishan.sanjeev,srinath.nair}@research.iiit.ac.in
radhika.mamidi@iiit.ac.in

Abstract
Sentiment analysis aims to detect the overall sentiment, i.e., the polarity of a sentence, paragraph, or text span, without considering the entities mentioned and their aspects. Aspect-based sentiment analysis aims to extract the aspects of the given target entities and their respective sentiments. Prior works formulate this as a sequence tagging problem or solve this task using a span-based extract-then-classify framework where first all the opinion targets are extracted from the sentence, and then with the help of span representations, the targets are classified as positive, negative, or neutral. The sequence tagging problem suffers from issues like sentiment inconsistency and colossal search space. Whereas, Span-based extract-then-classify framework suffers from issues such as half-word coverage and overlapping spans. To overcome this, we propose a similar span-based extract-then-classify framework with a novel and improved heuristic. Experiments on the three benchmark datasets (Restaurant14, Laptop14, Restaurant15) show our model consistently outperforms the current state-of-the-art. Moreover, we also present a novel supervised movie reviews dataset (Movie20) and a pseudo-labeled movie reviews dataset (moviesLarge) made explicitly for this task in English language and report the results on the novel Movie20 dataset as well.

1 Introduction
Online reviews and tweets play an essential role in consumer decision making. Hence, it becomes crucial to efficiently and effectively extract user opinions from an unstructured text (user review, tweet). Aspect-Based Sentiment Analysis is a fundamental task in mining opinions and sentiment analysis (Pang and Lee, 2008; Liu, 2012). This task requires detecting the aspects of the target entities mentioned (Aspect Extraction) and detecting the sentiment attached, i.e., the polarity of the target entity (Sentiment Classification). Hence, it is more challenging than the traditional sentence-level sentiment analysis (Lin and He, 2009; Kim, 2014), where we predict the text’s polarity as a whole. As shown in Table 1, given a sentence, the task is to extract the aspects “acting” and “editing” and predict the corresponding polarity, which is positive, negative, respectively.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Great acting dreadful editing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Targets</td>
<td>[acting], [editing]</td>
</tr>
<tr>
<td>Polarities</td>
<td>Positive, Negative</td>
</tr>
</tbody>
</table>

Table 1: Example Sentence

Most of the previous works treat this Aspect-Based Sentiment Analysis (ABSA) task as a combination of two different subtasks, namely, Aspect Extraction (AE) and Sentiment Classification (SC). Researchers often treat these two subtasks as independent and work on both of them individually, or some even tried to combine the two and propose a joint model that can extract the targets and predict the polarity. Much work has been done in the field of AE. Jakob and Gurevych (2010); Liu et al. (2015); Wang et al. (2016a); Poria et al. (2016); Shu et al. (2017); He et al. (2017); Xue and Li (2018); Fan et al., (2018) formulate AE as a sequence tagging problem. In sequence tagging, the task is to mark each word with a set of tags (e.g. B, I, O). The second subtask, SC, i.e., marking each extracted target term as Positive, Negative, or Neutral, has also been widely studied (Tang et al., 2016; Wang et al., 2016b; Chen et al., 2017; Xue and Li, 2018; Fan et al., 2018). The main issue with most of these sentiment classifiers is that, they assume the target is already given.

Zhang et al. (2015) and Li et al. (2019) tried to combine the two subtasks and solve the task in a more integrated way by jointly extracting targets and predicting their sentiments. The main idea
here is either jointly marking the words with a set of tags for the task of AE (e.g., B, I, O) and also marking them as Positive, Negative, Neutral for SC or use a more collapsed version of marking (e.g., B-Sentiment, I-Sentiment, O).

There are many disadvantages to the above BIO annotation scheme. As shown by Lee et al. (2016), using BIO tags for extractive question answering (in our case extracting opinion terms) have issues like colossal search space since the model must consider the power set of all words in a sentence. This results in it being less effective, and in the case of polarity classification, sequence tagging is not optimal because tagging the polarity over each word fails to capture the semantics of the entire opinion-target. Moreover, there could be sentiment inconsistencies in a multi-word target-term as predicted polarities over different words in a target could be different.

In this paper, we make the following contributions:

1. We propose TEASER, a span-based labeling scheme methodology exploiting the extract-then-classify framework for aspect-based sentiment analysis that reduces the search space while dealing with half-word coverage issues and overlapping spans.

2. We conduct extensive experiments that demonstrate our model to consistently outperform the current state-of-the-art in Aspect Extraction and overall ABSA task on the three benchmark datasets (Restaurant14, Restaurant15, Laptop14).

3. We also present two novel datasets\(^{1}\) in the domain of movie-reviews. The Movie20 dataset is a Supervised dataset of 1162 sentences made explicitly for this task, and movies-Large is a Pseudo-labeled dataset of 14373 sentences.

2 Related Work

Hu and Liu (2004) proposed Aspect-based Sentiment Analysis for the first time. Since then, it has been widely studied, especially in recent years (Zhang et al., 2018).

Most existing works treat this as a combination of Aspect Extraction and Sentiment Classification.

A joint modeling approach for AE and SC was proposed by Goldie et al. (2019). They used a BiLSTM with attention mechanism to capture the relationship between the two tasks.

For AE, there are various methods tried and tested for this task. Jakob and Gurevych (2010); Wang et al. (2016a); Shu et al. (2017) made use of Conditional Random Fields (CRFs) for prediction. Poria et al. (2016) tried a deep learning approach for this task of AE for the first time. They proposed a CNN based model to tag words in sentences as an aspect or non-aspect. Xu et al. (2018) Used a double embedding Mechanism with Convolutional Neural Networks (CNNs) to solve this task. Liu et al. (2015) tried to tackle this with Recurrent Neural Networks (RNNs). Similar to AE, the subtask of SC has been widely studied (Jiang et al., 2011; Vo and Zhang, 2015; Wang et al., 2018; Zhu and Qian, 2018; Chen and Qian, 2019; Zhu et al., 2019). Dong et al. (2014) proposed an RNN based approach, Chen and Qian (2019) attempted to solve this using Transfer Capsule Network, and Li and Lam (2017) used Memory networks.

Few works tried to propose a joint (unified) model for both the tasks of AE and SC. There are mainly two ways: Joint training and Collapsed tagging. In the former, a multi-task learning framework is built where both the subtasks, AE and SC have individual tags and are trained independently, and they may have some shared features. Then the two models are combined during inference. Meanwhile, in the latter, a collapsed set of tags e.g. B-Sentiment, I-Sentiment, O are used, and then a single model is trained combinedly for both the tasks.

Mitchell et al. (2013) formulated this task as a sequence tagging model and proposed a model using CRFs for the same. Li et al. (2019) made use of the collapsed tagging scheme, involving two stacked RNNs and a gate mechanism to maintain sentiment consistency. Zhou et al. (2019) proposed a span-based joint model using BiLSTMs and an attention mechanism to compute the sentiment information towards each span. Hu et al. (2019) proposed a pipelined span-based extract-then-classify framework using BERT (Devlin et al., 2018) as a backbone network jointly trained on AE and SC. Chen and Qian (2020) tried to exploit the interactive relation between the two subtasks by constructing a multi-layer multi-task framework with a relation propagation mechanism and thereby boosting the performance of both the subtasks.

\(^{1}\)The datasets can be found here [https://github.com/vaibhavb26/Movie-Reviews-Datasets](https://github.com/vaibhavb26/Movie-Reviews-Datasets)
<table>
<thead>
<tr>
<th>Sentence</th>
<th>Great acting dreadful editing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipeline</td>
<td></td>
</tr>
<tr>
<td>Target Start: 2, 4</td>
<td>Target End: 2, 4</td>
</tr>
<tr>
<td>Polarity: +, -</td>
<td></td>
</tr>
<tr>
<td>Collapsed</td>
<td></td>
</tr>
<tr>
<td>Target Start: 2+, 4-</td>
<td>Target End: 2+, 4-</td>
</tr>
</tbody>
</table>

Table 2: Span-based labeling scheme

<table>
<thead>
<tr>
<th>Sentence</th>
<th>The service was exceptional - sometime there was a feeling that we were served by the army of friendly waiters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Aspects</td>
<td>[service], [served], [waiter], [waiters]</td>
</tr>
<tr>
<td>Gold</td>
<td>[service], [waiters]</td>
</tr>
</tbody>
</table>

Table 3: Half Word Coverage and Overlapping spans

3 Methodology

3.1 Preliminaries

Hu et al. (2019) proposed a span-based labeling scheme, as shown in Table 2, i.e., annotating each opinion target with its span boundary followed by its sentiment polarity. While this model reduces the search space marginally, the approach has issues like overlapping spans, half-word coverage.

As shown in the example in Table 3, the sub-word [waiter] is being predicted twice; i.e., it is part of two different predicted spans ([waiter] and [waiters]). This is a case of overlapping spans as each sub-word should be a part of no more than one span, and since this output will then be sent to the sentiment classifier, there could be a problem of sentiment inconsistency for the word “waiters”. Also, half-word coverage is evident here as “waiter” should not be considered as an aspect, instead, “waiters” is more appropriate, because if “waiter” is considered as an aspect it will lead to having two different tags for the word “waiters” (one tag for “waiter” and one tag for “s”) which is incorrect. Though the work clearly states that they remove redundant spans with the word-level F1 function but since BertTokenizer tokenizes the words into sub-words (e.g. Waiters being tokenized to “waiter” and “s”) the redundancy issue persists.

3.2 Supervised Method

We formulate ABSA in a different way as compared to most of the previous works which treat ABSA as a sequence-tagging problem. As shown by (Lee et al., 2016), it is more beneficial to predict the two endpoints of a span as compared to sequence-tagging (BIO prediction). Hence, we use a similar approach of predicting the two endpoints. Similar to (Hu et al., 2019), we make use of span-based labeling scheme which is as follows: given an input sentence \( x \) of length \( n \) i.e. \( x = \{x_1, x_2, \ldots, x_n\} \), we make three different lists, \( \text{start\_positions} \), \( \text{end\_positions} \) and \( \text{polarities} \), each of length \( m \) where \( m \) is the number of targets in the sentence. \( \text{start\_positions} \) is a list containing the start position of each target in a sentence. Similarly, \( \text{end\_positions} \) is a list containing the end position of each target and \( \text{polarities} \) is the list containing the polarities (Positive, Negative, Neutral) of each target.

We build two different models for the two sub-tasks of Aspect Extraction and Sentiment Classification. These two models are separately trained and combined as a pipeline during inference. The pre-trained BERT model can be finetuned with just one additional layer to create state-of-the-art results in many tasks (Devlin et al., 2018). Therefore we use the BERT encoder as the primary network in both the subtasks. Using the pre-trained transformer blocks (Vaswani et al., 2017), the word embeddings are mapped to contextualized token representations. An Aspect extractor is used to extract the multiple possible targets from the sentence. Then, a polarity classifier (Hu et al., 2019) is used to predict the polarity of each extracted target using the summarized span representation.

3.2.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) can achieve state-of-the-art results in a lot of NLP tasks (Devlin et al., 2018) and hence we use it as our main network. Given a sentence \( x \), we first tokenize the sentence using BertTokenizer(based on wordpiece) with a vocabulary of 30522 tokens. Then we put a [CLS] token at the start and [SEP] token at the end of the tokenized sentence to form a new sentence \( y \) of length \( n + 2 \)
(considering \( n \) as the length of \( y \)). For each token \( y_i \in y \), its input representation is constructed by summing the corresponding token, segment and position embeddings (Devlin et al., 2018). Now the input representation is passed to the series of \( L \) stacked transformers blocks(\( L = 12 \) for BERT-base and \( L = 24 \) for BERT-Large) to get the contextual representations. It has been used in various downstream tasks including GLUE (Devlin et al., 2018), subjective bias detection (Pant et al., 2020), and sarcasm detection (Pant and Dadu, 2020). We suggest readers go through (Devlin et al., 2018) and (Vaswani et al., 2017) to get an in-depth understanding of BERT and the transformer block architecture, respectively.

### 3.2.2 Aspect Extractor

The aim of Aspect extractor is to extract all possible opinion targets from a given sentence. Instead of tagging the sentence sequentially, we detect the target by predicting the start and end positions of the targets (Hu et al., 2019). We add another layer on top of the BERT model. Using this, we get the confidence score for the start and end position as shown in Equation 1, where \( h \) is the contextual representation of the input (Output of BERT) and \( w_s, w_e \) are trainable weight vectors.

\[
\begin{align*}
    c_s &= w_s h, \quad p_s = softmax(c_s) \\
    c_e &= w_e h, \quad p_e = softmax(c_e)
\end{align*}
\]

(1)

For training, we then generate two lists, a list of \( \text{starts} \) and a list of \( \text{ends} \), each of length \( n + 2 \). Each position in the \( \text{starts} \) signifies if any span in the training sentence starts at the given position. Similarly, the list of \( \text{ends} \) signifies if any span in the training sentence ends at the given position. And the probabilities \( p_s \) and \( p_e \) are calculated as shown in Equation 1. The training objective is the sum of the negative log probabilities of the true start and end positions on the two predicted probabilities (Hu et al., 2019). The training objective is shown in Equation 2 where \( y_s \) and \( y_e \in \mathbb{R}^{n+2} \), and each element \( y_s^i \) indicates whether the \( i \)-th token starts a target and \( y_e^i \) indicates whether the \( j \)-th token ends a target(Hu et al., 2019).

\[
L = - \sum_{i=1}^{n+2} y_s^i \log(p_s^i) - \sum_{j=1}^{n+2} y_e^j \log(p_e^j)
\]

(2)

Once we get the confidence scores \( c_s \) and \( c_e \), the objective is to choose the non-overlapping spans ensuring no half-word coverage that has maximum value of \( c_s^i + c_e^j \) such that \( i <= j \). As shown by (Hu et al., 2019), choosing the top \( k \) spans could lead to overlapping spans. Hence, we present a TEASER’s heuristic for Aspect Extraction as shown in Algorithm 1. The algorithm helps remove the overlaps as well as half-word coverage. Firstly, we choose top \( M \) indices from both the confidence scores\( (\text{starts}, \text{ends}) \). For each pair \( s_i, e_j \) such that \( s_i \in \text{starts} \) and \( e_j \in \text{ends} \), \( s_i <= e_j \) and \( s_i \) is a start of a word and \( e_j \) is an end of a word, The heuristic score is defined as \( c_s^i + c_e^j - \sqrt{\text{length of the target}} \). It is inter-

**Algorithm 1: TEASER’s Heuristic for Aspect Extraction**

**Input:** \( c_s, c_e, \alpha, K \)

\( /* c_s: \) score of start position */
\( /* c_e: \) score of end position */
\( /* \alpha: \) threshold value */
\( /* K: \) maximum proposed targets */

1. \( P, \text{Out}, H = \{\}, \{\}, \{\} \)
\( /* P: \) preliminary predictions */
\( /* \text{Out}: \) output list */
\( /* H: \) heuristic score */

2. \( \text{selected} = \{\} \)

3. \( \text{starts}, \text{ends} = \text{Top-M indices of } c_s, c_e \)

4. \( \text{for } s_i \in \text{starts} \text{ do} \)

5. \( \text{for } e_j \in \text{ends do} \)

6. \( \text{if } s_i < e_j \text{ and } c_s[i] + c_e[j] \geq \alpha \) then

7. \( \text{target} = [s_i, e_j] \)

8. \( \text{score} = c_s[i] + c_e[j] - \sqrt{\text{length of target} + 1} \)

9. \( P = P \cup \text{target} \)

10. \( H = H \cup \text{score} \)

11. \( P.\text{sort()} /* \) sort based on Heuristic score in reverse order */

12. \( \text{for } \text{pred} \in P \text{ do} \)

13. \( \text{if } \text{size}(\text{Out}) < K \text{ then} \)

14. \( s_i, e_i = \text{pred.start.position}, \text{pred.end.position} \)

15. \( \text{if } \forall i \in \{s_i, e_i\} \notin \text{selected} \) then

16. \( \text{Out} = \text{Out} \cup \text{pred} \)

17. \( \text{selected} = \text{selected} \cup [s_i, e_i] \)

18. else

19. \( \text{break} \)

20. \( \text{return Out} \)
testing to note that, the heuristic score is a function of the length of the target and this is very important for the performance of the model as the targets are usually short entities. If the heuristic score of these two indices is greater than a certain threshold (manually tuned), we add it to the list of preliminary predictions. preliminary predictions is a list of predictions which follow the heuristic condition but it also has overlapping targets. To remove the overlaps we maintain another list of selected tokens, selected, which helps in identifying if a token was a part of a better prediction.

We sort the preliminary predictions in reverse order with the most confident prediction being in the first place and so on. We then iterate through the list, let the start position and end position of the current prediction we are looking at be \( t_s, t_e \) respectively. If any token \( e \in [t_s, t_e] \) was already present in a previous prediction (which is calculated using the selected list), we discard the current prediction. If no token is present in the selected list, we add the prediction to the list of targets and mark all tokens \( e \in [t_s, t_e] \) as selected.

We repeat this until we reach the end of preliminary predictions or the maximum number of targets are extracted. The pseudocode of the algorithm is as shown in Algorithm 1.

### 3.2.3 Polarity Classifier

Instead of using sequence tagging methods, we calculate a summarized vector from the contextualized sentence vectors according to the span boundary (Hu et al., 2019). The summarized vector is calculated using attention mechanism (Bahdanau et al., 2015) and the sentiment polarity is predicted with the help of feed-forward neural networks.

We obtain the polarity score by applying a linear transformation followed by a Tanh activation and another linear transformation which is then normalized using the softmax function as shown by (Hu et al., 2019).

\[
\begin{align*}
g^p &= W_p \tanh(W_v v) \\
p^p &= \text{softmax}(g^p)
\end{align*}
\]

where \( W_v \in \mathbb{R}^{h \times h} \) and \( W_p \in \mathbb{R}^{k \times h} \) are two trainable parameter matrices.

We minimize the negative log probabilities of the true polarity on the predicted probability. We calculate the polarity probability for each candidate target span present in the set \( \mathbf{O} \) during inference and choose the sentiment class with the highest \( p^p \).

## 4 Semi-Supervised Learning

### 4.1 Dataset Creation

We scrape the 15535 sentences from 3200 movie reviews from a leading movie review website. Users rate a movie on a scale of 1 (very bad) to 10 (very good). To avoid any potential bias, we chose the most popular movies with the most reviews, and the reviews were chosen uniformly on a rating scale of 1 to 10. We then divide the dataset into two parts: 14373 sentences (moviesLarge) for training the semi-supervised model and 1162 sentences (Movie20) to validate the semi-supervised model.

Two human annotators with proficiency in English and linguistic background performed the annotation of the dataset’s validation split (Movie20). The annotation was performed according to the original guidelines as set in (Pontiki et al., 2014b) on the following aspects:

1. **Opinion Targets**: Given a sentence, identify all the aspect terms present in the sentence. e.g., “Stunning visuals, amazing storyline.” The aspect terms in the sentence are “visu- als”, “storyline”.

2. **Target Sentiment**: Assuming that we know the aspect terms beforehand, determine the sentiment attached, i.e., the polarity of each term (Positive, Negative, Neutral). e.g., visuals - Positive, storyline - Positive (Considering the example mentioned above.)

Table 4 shows a few example instances from the novel Movie20 dataset.

For validating the quality of the annotation process, we use the Inter-Annotator Agreement of both the tasks through Cohen’s Kappa Coefficient (Fleiss and Cohen, 1973). We obtain a Kappa score of 0.8326 for the annotation process. The Kappa score implies that the annotation process is of high quality, with the annotators showing a high degree of agreement.

### 4.2 Pseudo-Labeling

The moviesLarge dataset has 14373 sentences. Since human annotation to such a large dataset is very time-consuming and complex, we use the Pseudo-Labeling technique. In Pseudo-Labeling, instead of manually annotating the dataset, we approximate labels to the dataset based on available labeled data.
It has very good cinematography shots and it is very entertaining. [cinematography shots] POS

I like it but the beginning was very long and slow while the end was all over the place trying to explain everything. [beginning], [end] NEG, NEG

It is action packed, fantasy filled and thoroughly exciting. [action], [fantasy] NEU, NEU

Great Acting dreadful editing. [Acting], [editing] POS, NEG

the whole cinematic experience is not there. [cinematic experience] NEG

Table 4: Example instances from Movie20 dataset.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Restaurant14</th>
<th>Restaurant15</th>
<th>Laptop14</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Test</td>
<td>Train</td>
</tr>
<tr>
<td>#Sentences</td>
<td>3040</td>
<td>800</td>
<td>1313</td>
</tr>
<tr>
<td>#Aspects</td>
<td>3603</td>
<td>1122</td>
<td>1209</td>
</tr>
</tbody>
</table>

Table 5: The statistics of the three datasets

Figure 1 shows the Pipeline of Semi-Supervised Learning. As shown, it can be divided into two parts: Pseudo-Labeling (1a) and final Model Training (1b). In case of pseudo-labeling, we combine all the three existing datasets, Laptop14, Restaurant14, and Restaurant15 and using this as our training data, we train our model TEASER. We then make use of the trained model to predict the labels of the unlabeled moviesLarge dataset. Since these labels aren’t manually annotated, these are the approximate labels and hence the process is called pseudo-labeling.

Once we have the pseudo-labeled moviesLarge dataset, we combine moviesLarge and the labeled datasets Laptop14, Restaurant14, and Restaurant15 and train the model using this as our training data. Finally, we test this model on our novel Movie20 dataset and derive the results. The details are discussed in subsection 6.2.

5 Experiments

5.1 Datasets

For all the experiments, we use the three benchmark datasets from various domains. The datasets were taken from SemEval 2014 (Pontiki et al., 2014b) and SemEval 2015 (Pontiki et al., 2014a) tasks which include 2 datasets from restaurant domain, Restaurant14 and Restaurant15. Moreover, we use another dataset Laptop14 made by using customer reviews from the laptop domain. The statistics of the datasets are as shown in the Table 5.

5.2 Metrics

We use the F1 score as the evaluation metric for the ABSA task. To analyze our TEASER’s heuristic for Aspect Extraction’s performance, we also compare using an F1 score between various models for the subtask of Aspect Extraction.

5.3 Experimental Settings

We use the pre-trained BERT-Large model for all the experiments. It has 24 layers (transformer blocks), 16 attention heads. For more details about BERT-Large parameters, readers can refer to (Devlin et al., 2018). The batch size is 32, \( M \) (number of candidate spans) is set to 20, while \( K \), the maximum number of proposed targets is 10, and the threshold is manually tuned. We use Adam optimizer with a learning rate of 2e-5.

\( ^2 \)We do not use the erroneous rest, total dataset as prescribed by the authors of Li et al. (2019).
Table 6: Results of the Aspect Extraction Task and the Aspect-based Sentiment Analysis task.

<table>
<thead>
<tr>
<th>Model</th>
<th>Restaurant14 F1 score</th>
<th>Restaurant15 F1 score</th>
<th>Laptop14 F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AE</td>
<td>ABSA</td>
<td>AE</td>
</tr>
<tr>
<td>MNN</td>
<td>83.05</td>
<td>63.87</td>
<td>70.24</td>
</tr>
<tr>
<td>E2E-TBSA</td>
<td>83.92</td>
<td>66.60</td>
<td>69.40</td>
</tr>
<tr>
<td>SpanABSA</td>
<td>86.71</td>
<td>73.68</td>
<td>74.63</td>
</tr>
<tr>
<td>DE-CNN</td>
<td>82.79</td>
<td>-</td>
<td>68.52</td>
</tr>
<tr>
<td>RACL-BERT</td>
<td>86.38</td>
<td>75.42</td>
<td>73.99</td>
</tr>
<tr>
<td>TEASER</td>
<td>88.76</td>
<td>75.53</td>
<td>79.78</td>
</tr>
</tbody>
</table>

Table 6: Results of the Aspect Extraction Task and the Aspect-based Sentiment Analysis task.

5.4 Baseline Methods

We compare our proposed model with the following approaches:

1. MNN (Wang et al., 2018) - This work proposed a unified (collapsed) tagging scheme for both the tasks of Aspect Extraction and Sentiment Classification.

2. SpanABSA (Hu et al., 2019) - It is a pipelined model with a multi-target extractor and a polarity classifier. It uses BERT-Large as the backbone network for both the subtasks.

3. E2E-TBSA (Li et al., 2019) - It has two stacked RNNs(Recurrent Neural Networks) with multi-task learning over a collapsed tagging scheme.

4. DE-CNN (Xu et al., 2018) - It is a model exclusively for Aspect Extraction, which uses a double embedding mechanism with CNNs(Convolutional Neural Networks).

5. RACL-BERT (Chen and Qian, 2020) - This is the current state-of-the-art method that proposes a Relation Aware Collaborative Learning (RACL) framework which allows the subtasks to work coordinately via the multi-task learning and relation propagation mechanisms in a stacked multi-layer network.

6 Results

6.1 Supervised Model

The Table 6 shows the comparison for all the methods. For the task of Aspect Extraction, our model achieves 2.05%, 5.15%, 4.82% absolute gains over the three benchmark datasets, which proves the efficacy of our model. Also, for the overall ABSA task, our model achieves 0.09%, 1.29%, 5.43% absolute gains, which is significant. The AE results prove that span-based extraction performs better than any of the other methods proposed. The overall ABSA results suggest that it is better to use two different models for the two subtasks and then combine via pipeline over jointly learning to predict them simultaneously. This further concretizes the fact mentioned by (Hu et al., 2019) that Target Extraction and Sentiment Classification are loosely coupled, i.e., there is a weak connection between them.

6.2 Semi-Supervised Model

As shown in Table 7, we report the Precision, Recall, and F1-score of the model on our novel Movie20 dataset for the AE and ABSA task. The F1 score for AE is 63.74% and 58.23% for ABSA. Through this, we set a strong benchmark for semi-supervised aspect-based sentiment analysis on a movie-based dataset. We further analyze the model’s predictions and discover the following patterns in the errors made by the model:

1. The model usually failed to mark aspects preceded by rare adjectives (i.e., adjectives that occurred in the dataset with less frequency). For example, in the following sentence, “Great Acting dreadful editing”, the words “acting” and “editing” are the targets with polarities Positive and Negative respectively. However, the model recognized “acting” but failed to recognize “editing” because the word “dreadful” occurs very rarely in the dataset. Fewer examples in the dataset could have caused the model to fail in such cases.

2. The model also failed to mark sentences where specific experiential knowledge about a movie’s good and bad aspects was required (usually in the absence of any clear adjec-
<table>
<thead>
<tr>
<th>Model</th>
<th>Movie20 Precision</th>
<th>Movie20 Recall</th>
<th>Movie20 F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AE</td>
<td>ABSA</td>
<td>AE</td>
</tr>
<tr>
<td>TEASER</td>
<td>81.91</td>
<td>79.91</td>
<td>52.17</td>
</tr>
</tbody>
</table>

Table 7: Results of the Semi-supervised Aspect Extraction Task and the Aspect-based Sentiment Analysis task.

tives). For example, in the following sentence, “We know the bliss can’t last. Thus, tears stream down your face during the third act”; the phrase “third act” should be marked as an aspect with a positive sentiment since movies that can connect with the audience’s emotions are considered good. However, the model does not have this experiential knowledge and hence failed to recognize the aspect. A similar example is the following sentence, “I could not relate to any character and did not care about the outcome.”, where the model failed to mark “character” as an aspect with negative sentiment since the model does not have the experiential knowledge that relatable characters make for a good movie.

7 Conclusion

In this work, we proposed TEASER, an extract-then-classify network for Aspect-based Sentiment Analysis with pre-trained BERT-Large as the main network. We also presented an Aspect extractor with a novel heuristic, which helps extract all the targets of a given sentence. Experiments show that our method consistently outperforms the current state-of-the-art in the task of AE and also in ABSA. We also presented two datasets, Movie20, a supervised dataset of 1162 sentences with a Cohen Kappa Score of 0.8326, and moviesLarge, a pseudo-labeled dataset of around 14373 sentences. Lastly, using Semi-supervised learning, we benchmarked TEASER on the Movie20 dataset. We analyzed the model to reason where the model failed to perform, and according to the findings, aspects preceded with rare adjectives and aspects with an absence of a clear adjective were the primary reasons for the failure.

References


Zhuang Chen and T. Qian. 2019. Transfer capsule network for aspect level sentiment classification. In ACL.


Li Dong, F. Wei, Chuanqi Tan, Duyu Tang, M. Zhou, and K. Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In ACL.

Feifan Fan, Yansong Feng, and Dongyan Zhao. 2018. Multi-grained attention network for aspect-level sentiment classification. In EMNLP.


Minghao Hu, Y. Peng, Z. Huang, D. Li, and Yiwei Lv. 2019. Open-domain targeted sentiment analysis via span-based extraction and classification. In ACL.


Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single and cross-domain setting with conditional random fields. In EMNLP, pages 1035–1045. ACL.


Xin Li, Lidong Bing, Piji Li, and W. Lam. 2019. A unified model for opinion target extraction and target sentiment prediction. In AAAI.

Xin Li and W. Lam. 2017. Deep multi-task learning for aspect term extraction with memory interaction. In EMNLP.


Kartikey Pant and Tanvi Dadu. 2020. Sarcasm detection using context separators in online discourse. In FIGLANG.


Lei Shu, Hu Xu, and Bing Liu. 2017. Lifelong learning crf for supervised aspect extraction. In ACL.

Interactive Learning Approach for Arabic Target-Based Sentiment Analysis

Husamelddin A.M.N Balla, Marisa Llorens Salvador and Sarah Jane Delany
School of Computer Science
Technological University Dublin
Dublin, Ireland
{husamelddin.balla,marisa.llorens,sarahjane.delany}@tudublin.ie

Abstract

Recently, the majority of sentiment analysis researchers focus on target-based sentiment analysis because it delivers in-depth analysis with more accurate results as compared to traditional sentiment analysis. In this paper, we propose an interactive learning approach to tackle a target-based sentiment analysis task for the Arabic language. The proposed IA-LSTM model uses an interactive attention-based mechanism to force the model to focus on different parts (targets) of a sentence. We investigate the ability to use targets, right and left contexts, and model them separately to learn their own representations via interactive modeling. We evaluated our model on two different datasets: Arabic hotel review and Arabic book review datasets. The results demonstrate the effectiveness of using this interactive modeling technique for the Arabic target-based sentiment analysis task. The model obtained accuracy values of 83.10 compared to SOTA models such as AB-LSTM-PC which obtained 82.60 for the same dataset.

1 Introduction

Sentiment analysis (SA) is one of the most prolific research areas in computer sciences, which aims to identify and extract user opinions from reviews. This technique has become an essential part of a wide range of applications in the areas of politics, business, advertising and marketing as it can help in identifying people's opinions towards related targets (Tang et al., 2015). Arabic is considered among the top 4 languages in terms of internet usage (Boudad et al., 2018). With the rapid growth of Arabic web content and low resources for analyzing Arabic opinion mining, the need for accurate Arabic sentiment analysis tools is very necessary.

There are three main classification levels in sentiment analysis: document-level, sentence-level, and target-level. Document-level SA aims to classify the sentiment expressed in the whole document. It considers the whole document as a basic information unit (talking about one topic). Sentence-level SA aims to classify the sentiment expressed in each sentence. In traditional sentiment analysis, the detailed opinions of all targets of the entity (which are required in many applications) are not provided. To acquire these details, we need to use the target level. Target-based sentiment analysis (TBSA) aims to classify the sentiment with respect to the specific targets of entities. The opinion holders can give different opinions for different targets of the same entity, like in this sentence: “The hotel is clean with good services, but the room was too small”. Target-based sentiment analysis is a fine-grained task in sentiment analysis. This kind of fine-grained target-based analysis generally relies on machine learning techniques that require large domain-specific datasets with manual training data (Hu and Liu, 2004).

Target-based sentiment analysis has become more popular in recent research as it delivers more accurate results compared to traditional sentiment analysis. Given a plain text, the trained model is able to detect the targets that were seen in the training set; the context is simply the sequence of words or tokens around the targets. Referring to the previous example, Target1: hotel, Context: clean, good service, Polarity: positive; Target2: room, Context: too small, Polarity: negative. In traditional sentiment analysis (non-target-based SA), this detailed level of analysis is not possible as the analysis is performed at sentence level and hence the entire sentence is classified as either positive or negative.

The research area of Arabic SA is relatively new. Recently, the work on Arabic SA has received a lot of attention and a number of papers on traditional SA have been published in the last couple of years (Shoukry and Rafea, 2012; Duwairi and El-Orfali, 2014; Nabil et al., 2015; Al-Rubaiee et al., 2016). However, research work on Arabic TBSA has not been addressed in depth yet. Most of the existing
TBSA research focuses on English (Xue and Li, 2018; Mowlaei et al., 2020; Ma et al., 2018) with very little work on other languages such as Arabic. In this paper, we address the task of TBSA in Arabic. The proposed model uses a neural network with an attention mechanism to force the model to attend to the important parts of a sentence. To achieve that, an interactive attention-based long short-term memory network (IA-LSTM) with an interaction technique is used to capture important information related to a given target.

Previous work in Arabic target-based SA used deep learning represented in Recurrent Neural Network (RNN) and LSTM models and developed several methods aiming to model contexts through the generation of target-based representations (Ruder et al., 2016; Tamchyna and Veselovská, 2016; Al-Smadi et al., 2019). However, the modeling of interactive targets with contexts using attention mechanisms was not addressed.

The model proposed in this paper (IA-LSTM) is based on a model (IAN) proposed for English language TBSA in a previously published paper (Ma et al., 2017) that addresses the separately modeled targets and contexts that jointly interact with each other with an attention mechanism. However, the left context was ignored in the previously proposed approach in (Ma et al., 2017). Given the fact that Arabic language is written from right to left, the addition of the left context may improve the performance as the most of the opinionated words come after the target (left context). The proposed model (IA-LSTM) is obtained by adding the left context to the IAN model from (Ma et al., 2017), to end up with a model that considers the three main elements (left context, target, and right context). Code for the proposed model is publicly available on Github 1.

The rest of this paper is organized as follows: section 2 presents the related work of SA and TBSA in English and Arabic. Section 3 discusses the methodology used for the proposed model in this paper. Section 4 describes the datasets used for evaluations. Section 5 explains the baselines, evaluations, the experimental results and analysis. Section 6 concludes this paper.

2 Related Work

The research work on Arabic TBSA has few examples as the published papers are limited, so, the first subsection of related work mainly focus on traditional SA approaches. While the second subsection covers the published work on TBSA on both Arabic and other languages.

2.1 Arabic SA

The early attempts on Arabic SA relied on the methods applied for English SA as it is more mature with rich resources in terms of SA. An example of this, Ahmad (2006) employed a rule-based approach that was originally designed for English. Almas and Ahmad (2007) modified the approach to accept other languages such as English and Urdu. Those two attempts used financial news datasets. The results from these experiments were similar across all of languages tested.

Continuing in a business-oriented and rule-based approach, Elhawary and Elfeky (2010) performed experiments on Arabic SA. The authors addressed the problem of SA using large data and MapReduce in an attempt to enhance the performance. Another work following a similar approach was proposed by Farra et al. (2010), which used a pre-compiled lexicon to improve the performance on both sentence-level and document-level SA. However, there were no significant improvements in general performance, except at sentence level SA with slight improvement.

Work on Arabic SA or any other language always requires datasets to test the used approach. As Arabic is a language with low resources, very few datasets have been created for Arabic SA over the past years. AWATIF Abdul-Mageed and Diab (2012) is one example of the existing datasets.

2.2 Target Based Sentiment Analysis (TBSA)

One of the earliest papers in TBSA was for English, and used a frequency-based approach proposed by Hu and Liu (2004). The basic idea behind this approach is counting the nouns in the text and considering the most frequently mentioned ones as targets. The authors also tried to avoid the error of incorrectly identifying infrequent nouns as targets by using the nearest opinion words. This approach was also used in several other papers (Qiu et al., 2009; Zhuang et al., 2006). To enhance this approach, Popescu and Etzioni (2007) proposed using a technique named “part of relationship” to eliminate the frequent nouns that are incorrectly identified as targets.

TBSA is more challenging than the traditional task of sentiment analysis (non-target-based) be-
cause the model needs to include the impact of the context words on the target. A deep learning approach in this task can be performed by representing context, generating a target representation, and then detecting the important parts of the sentence (i.e. the targets). Again, RNNs have proven their competitive performance in this task in terms of capturing long-term dependency in sentences and general semantic classification. Moreover, the best RNN performers are the ones that include attention or memory networks. This shows that the models can learn how to concentrate on different parts of the sentence with an attention weight aggregated from a lower level to classify targets and opinion words and the link between them. Several researchers adopted an attention mechanism for this task in English (Gers et al., 2000; Song et al., 2019; Li et al., 2018). In general, there are not many techniques using deep learning for Arabic sentiment analysis and, in particular, TBSA (Dahou et al., 2016; Ruder et al., 2016; Tamchyna and Veselovská, 2016; Al-Smadi et al., 2019).

The International Workshop on Semantic Evaluation (SemEval), one of the most significant events in natural language processing (NLP) research, is concerned with the evaluation of computational semantic analysis systems. A special event focused on TBSA was organized in 2014, 2015 and 2016. In this workshop, Al-Smadi et al. (2018) proposed Arabic TBSA paper which performs a comparison between a deep neural network and SVM models. The authors used an RNN framework named Deeplearning4j that provides a set of implementations for different deep neural network algorithms. They evaluated their models on the Arabic hotel review dataset (Mohammad et al., 2016). Their deep neural network model outperformed the SVM model in accuracy of sentiment polarity while SVM outperformed the deep learning model in target extraction. They also proposed another study (Mohammad et al., 2016), which was part of the SemEval-TBSA 2016 competition. In this study they created a hotel review dataset with baselines obtained by using the SVM model with only unigram feature.

Focusing on Arabic TBSA and using the same dataset that this research is using (Arabic hotel reviews), three papers were recently published (Ruder et al., 2016; Tamchyna and Veselovská, 2016; Al-Smadi et al., 2019). Ruder et al. (2016) proposed a deep learning-based approach (INSIGHT-1) for multi-lingual TBSA as one of SemEval-2016 participants, which used a convolutional neural network for target extraction and sentiment analysis. Using the Arabic hotel review dataset their model outperformed the other participants in the workshop. Another approach proposed by Tamchyna and Veselovská (2016) for the same task used an RNN-based binary classifier for the task of target-category identification. Their model was trained using word embedding features and achieved good performance as the second rank after (Ruder et al., 2016).

A recent attempt by Al-Smadi et al. (2019) used targets and context embeddings in their proposed model (AB-LSTM-PC). The approach models the context words via LSTM networks and then combines the word’s hidden states with target embeddings to generate the attention vectors. In addition, to further strengthen the effect of target embeddings, the model appends target embeddings, with each word embedding vector forming the context. This is used to produce the final representation for TBSA. Their model performance was the highest among all previously published papers evaluated on the same dataset we are using in this research.

An approach (IAN) for English TBSA using interactive targets and context representations was proposed in Ma et al. (2017). The model uses attention mechanisms to concatenate the separately modeled targets and context as final representation before it is fed to the softmax layer. This model only considers the right context of targets, ignoring the left context.

All the research mentioned in this section ignores the the left and right context of targets and the interactions between them. Using these extra features can increase the amount of information about the context by providing a more comprehensive approach to context. In this paper, we propose an interactive learning approach based on (Ma et al., 2017) to tackle sentiment polarity identification for the Arabic language which includes the use of left and right context.

3 Proposed Approach

Recurrent neural networks (RNNs) are deep learning neural networks designed specifically to learn sequences of data and are mainly used for textual data classification. The learning process is done at hidden recurrent nodes depending on their previous layers of nodes. However, RNNs suffer from the
vanishing gradient problem when handling long sequences of data. Bi-directional long short-term memory (Bi-LSTM) (Schuster and Paliwal, 1997) was proposed as a solution for this problem and have proven to be efficient in many NLP-related problems. In contrast to the standard RNNs, Bi-LSTM units have a major role in extracting and learning important features out of the input or computed data and keeping the computed values as long as they are needed in the memory vector.

In this research, we focus on target-based sentiment polarity classification considering the right and left contexts. The enhancement of sentiment classification performance can be achieved by considering the targets and their contexts. The good performance relies on simultaneously modeling targets and contexts precisely. Targets and contexts can influence the representation of each other. For example, the target word “hotel” can be naturally associated with the context word “clean” and vice versa. Therefore, targets and contexts can be modeled individually but learn from their interactions. In the input text, each word usually has its own contribution or importance, which is different from other words in the final representation for sentiment analysis. For instance, the importance of the word “room” is higher in the representation of the target “room price”, which is described by “expensive”. Therefore, in the proposed model, the attention weights for both targets and contexts are computed to respectively capture their important information.

The proposed model is an interactive attention LSTM-based (IA-LSTM) model which employs long short-term memory networks (LSTM) and attention mechanisms. To get important information from the left and right context, the model uses an attention mechanism associated with a target then computes context representation for sentiment polarity identification. In addition, the proposed model makes use of the interactive information from the word’s context to supervise the target modeling. Finally, the model concatenates the attended left context representations, target representations, and right context representations, then uses them to predict the sentiment polarity (see Fig. 1).

Following the model notation in Fig. 1, assume that a left context consists of k words \( [w_{LC1}, w_{LC2}, ..., w_{LCk}] \), a target consists of \( m \) words \( [w_1, w_2, ..., w_m] \), and a right context consists of \( n \) words \( [w_{RC1}, w_{RC2}, ..., w_{RCn}] \). We use pre-trained word embeddings for word representation of contexts and targets. Then, since there is a strong dependence between words in a sentence, the LSTM network is used to learn the hidden word semantics as LSTM is good at learning long-term dependencies. Next, LSTM produces the hidden states \( [h_{LC1}, h_{LC2}, ..., h_{LCk}] \) for the left context words, \( [h_1, h_2, ..., h_m] \) for the target words, and \( [h_{RC1}, h_{RC2}, ..., h_{RCn}] \) for the right context as word representations. Then, the model calculates the average of the hidden states to get the initial representations of the contexts and target.

The attention mechanism in the proposed model is adopted by using the initial representations of the contexts and target as input to help in selecting important information for classifying the sentiment polarity. To form the attention mechanism, we use the average as well as the last hidden state output of target and contexts to capture an abstract representations of the input sequence. The use of the average value is to form the initial representation of the other component (target or context) and combine it with the last hidden state of the current component. We found this is the best way to form the attention mechanism and it help to reduce the noise and the sparse information associated with the Arabic language (ElNagar et al., 2020).

The attention process is described by considering the left context, target, and the right context, as shown in Fig. 1. Using the left context words representation \( [h_{LC1}, h_{LC2}, ..., h_{LCk}] \) and the average of hidden states of target representation \( T_{avg} \), the model computes the left attention vector \( L_{Oi} \). Similarly, the model computes the target attention vector \( \beta_i \) by using the average of left context words \( L_{avg} \), the target words representation \( [h_1, h_2, ..., h_m] \), and the average of the right context vector \( R_{avg} \). The same technique is followed for the right attention vector \( R_{Oi} \) obtained using the right context words representation \( [h_{RC1}, h_{RC2}, ..., h_{RCn}] \) and the average of hidden states of target representation \( T_{avg} \). Then, the attention vectors \( L_{Oi}, \beta_i \), and \( R_{Oi} \) are used to obtain the word attention weights and concatenated as one vector before being fed to the softmax classifier.

Formally, given the word embeddings of the contexts and target, the hidden state representations of the contexts (left and right) and target are \( [h_{LC1}, h_{LC2}, ..., h_{LCk}], [h_{RC1}, h_{RC2}, ..., h_{RCn}] \) and \( [h_1, h_2, ..., h_m] \) respectively. The initial representation of contexts (left - equation (1) and right - equation (2)) and targets equation (3) is obtained by calculating...
the average of each as follows:

\[ LC_{\text{avg}} = \frac{1}{k} \sum_{i=1}^{k} h_{LC}^i \]  

(1)

\[ RC_{\text{avg}} = \frac{1}{n} \sum_{i=1}^{n} h_{RC}^i \]  

(2)

\[ T_{\text{avg}} = \frac{1}{m} \sum_{i=1}^{m} h_{T}^i \]  

(3)

To make the model focus on the important parts of the representations, an attention mechanism that employs the initial representations of the contexts and target is used. The target influence on the left context and the influence of the left context on the target is considered, as well as the influence of the target on the right context and the influence of the right context on the target. The left attention vector \( L\alpha_i \) is generated using both the left context and target representations as in equation (4):

\[ L\alpha_i = \frac{e^{\gamma(h_{LC}^i, T_{\text{avg}})}}{\sum_{j=1}^{k} e^{\gamma(h_{LC}^j, T_{\text{avg}})}} \]  

(4)

Where \( \gamma \) is a score function that computes the importance of \( h_{LC}^i \) and \( h_{RC}^i \) in the left and right context respectively. This function is defined as equation (5):

\[ \gamma(h_{LC}^j, T_{\text{avg}}) = \tanh(h_{LC}^j \cdot w \cdot T_{\text{avg}}^t + b) \]  

(5)

Where \( w \) is a weight matrix, \( b \) is a bias, and \( T_{\text{avg}}^t \) is a transpose of \( T_{\text{avg}} \). The right attention vector \( R\alpha_i \) is generated using both the right context and target representations as equation (6):

\[ R\alpha_i = \frac{e^{\gamma(h_{RC}^i, T_{\text{avg}})}}{\sum_{j=1}^{n} e^{\gamma(h_{RC}^j, T_{\text{avg}})}} \]  

(6)

Similarly, the target attention vector \( \beta_i \) is calculated using all of the right context, left context, and the target as (7):

\[ \beta_i = \frac{e^{\gamma(h_{T}^i, LC_{\text{avg}})}}{\sum_{j=1}^{m} e^{\gamma(h_{T}^j, LC_{\text{avg}})}} + \frac{e^{\gamma(h_{T}^i, RC_{\text{avg}})}}{\sum_{j=1}^{m} e^{\gamma(h_{T}^j, RC_{\text{avg}})}} \]  

(7)

The representations of the left context, target, and right context are computed as in equations (8), (9), (10) respectively:

\[ C_l = \sum_{i=1}^{k} L\alpha_i h_{LC}^i \]  

(8)
\[ T = \sum_{i=1}^{m} \beta_i h_{RC}^i h_{LC}^i \tag{9} \]
\[ C_t = \sum_{i=1}^{n} R\alpha_i h_{RC}^i \tag{10} \]

The final representation is done by concatenating the representations of the three components of the left context, target, and the right context (\( C_1 \), \( T \), and \( C_r \) respectively) as one vector \( (F) \). Finally, the prediction of sentiment polarity (positive, neutral, negative) is done by using the softmax layer as in equation (11):

\[ x = tanh(w_1 F + b_1) \tag{11} \]

Where \( w_1 \) is the weight matrix and \( b_1 \) is the bias.

### 3.1 Hyperparameters and Model Training

The pre-trained Arabic word embeddings “AraVec” (Soliman et al., 2017) was used for the target and the context word embeddings with a dimension of 300 nodes. AraVec is an open-source project which provides free-to-use Arabic word embeddings trained on more than 3 billion words from web pages and Wikipedia. A uniform \( \beta \) distribution \( U(0:1; 0:1) \) was used to initialize all out-of-vocabulary words and weights. The model employs the momentum optimization algorithm (Qian, 1999) to train the parameters, which adds a fraction of the update vector in the preceding step to the current update vector. The dropout rate is set to 0.5, and the normalization coefficient \( L2 \) in the objective function is set to \( 10^{-5} \).

### 4 Datasets

There are two benchmark datasets were used to evaluate the proposed approach, Arabic hotel reviews and Arabic book reviews datasets.

#### 4.1 Arabic Hotel Reviews

This dataset was part of the work proposed in task 5 of the Sem-Eval 2016 workshop on target-based sentiment analysis (Pontiki et al., 2016; Mohamad et al., 2016). The dataset contains the 24,028 TBSA annotated tuples provided (19,226 tuples for training and 4,802 tuples for testing). For the sake of generalization and to avoid the single dialect problem, the original dataset was collected from well-known different Hotels’ booking websites such as Booking.com, TripAdvisor.com. The selected reviews in the datasets belongs to Hotels from different Arabian cities in different countries such as Dubai, Beirut, Amman, Mecca, etc. In addition, the dataset was annotated on both text-level (2,291 reviews’ texts) and sentence-level (6,029 annotated sentences). In this research, we consider only the sentence-level tasks. This is a manually annotated dataset, whereas, for each sentence, a tuple of target category, opinion target expression, and target polarity were annotated. The sentiment polarity labels (positive, negative, neutral) were used to annotate the polarity of each target or category.

#### 4.2 Arabic Book Reviews

This dataset was provided by (Al-Smadi et al., 2015) as a benchmark Human Annotated Arabic Dataset (HAAD). HAAD is a book review dataset in Arabic, which has been constructed and annotated by humans, taking into account the target terms and their polarities. For each review sentence, a tuple consisting of an target-category, target-category polarity, target-term, and target-term polarity was extracted and annotated. A sentiment polarity (positive, negative, conflict, neutral) was used to represent both the target-category and target-term sentiment polarity based on the annotated sentences. This dataset consists of 1513 Arabic book reviews annotated with aspect terms, aspect term polarity, aspect category, and aspect category polarity. In our experiments, we considered the labels positive, negative, and neutral excluding conflict label.

Lastly, both datasets were proposed as three separated machine-readable XML format files annotated training, test, and gold test with polarity distribution in Table 1. In addition, one sentence can contains more than one target which can be assigned with different polarity label. SemEval-TBSA designed a specific way of evaluations for all of the models evaluated on the associated datasets. F1 score is usually used as a metric for category and target execration. However, the metric used for sentiment polarity in the literature review is the accuracy of each model. A sample of Arabic hotel reviews dataset is shown in Fig. 2.

### 5 Evaluation

We evaluated the proposed model on the Arabic hotel review and the Arabic book review. The evaluation is done by computing the accuracy of the polarity. The used accuracy metric defined as the
number of correctly predicted polarity labels of the (gold) targets, divided by the total number of the gold targets. The test set contains sentences without polarity labels which is expected to be predicted by the model and compared with the labels in the test gold set containing the same sentences. We compared the proposed model performance with the related work and SOTA approaches stated as LSTM (Gers et al., 2000), IAN (Ma et al., 2017), INSIGHT-1 (Ruder et al., 2016), and AB-LSTM-PC (Al-Smadi et al., 2019) evaluated on the same Arabic hotel review dataset. For the Arabic book review dataset, the proposed model is compared with a subset of these approaches: LSTM (Gers et al., 2000) and IAN (Ma et al., 2017). In addition, the baselines labelled as “Baseline” in Table 2 are the baselines provided with the published work of dataset creation. The baseline for the Arabic hotel review dataset was obtained by using SVM with N-grams as features. The baseline for the Arabic book reviews was obtained by using a frequency approach.

5.1 Compared Models

The models used in the comparison in this paper are as follows: (i) LSTM uses one LSTM network to model the context and get the hidden state of each word. The average value of all the hidden states is obtained as the final representation and fed to a softmax function to estimate the probability of each sentiment label (Gers et al., 2000). (ii) IAN uses an interactive attention-based LSTM considering the target and right context only (ignoring the left context) (Ma et al., 2017). (iii) INSIGHT-1 uses a convolutional neural network (CNN). The model concatenates the target vector with each word embedding and then applies a convolution over it to identify the sentiment polarity (Ruder et al., 2016). (iv) AB-LSTM-PC uses an attention-based LSTM by adding the target embedding in the input. To generate the attention vectors, the approach models the context using LSTM networks and combines the hidden states with the target embeddings (Al-Smadi et al., 2019).

For INSIGHT-1 and AB-LSTM-PC, we report the previously published results without implementing the models as they were evaluated on the same dataset. The rest of the compared models (LSTM and IAN) were implemented and evaluated on the datasets.

5.2 Experimental Results and Analysis

Table 2 shows the proposed model’s performance compared with other models. The worst performance in this table was for the LSTM model. This is most likely due to the fact that it does not make use of the attention mechanism, confirming findings in previous research (Jiang et al., 2011). Both the AB-LSTM-PC model (SOTA) and INSIGHT-1 have similar performance and outperform the LSTM model, confirming that the attention mechanism enhances the ability to identify sentiment polarity. Unlike IAN in (Ma et al., 2017), our proposed IA-LSTM model (represented by “IA-LSTM” in the table) takes a further step towards confirming the importance of considering the targets and contexts in the learning process interactively. As shown in Table 2, the IA-LSTM model achieved the highest performance, outperforming all baselines and the other approaches. This enhancement can be explained by the fact that our model uses three connected attention networks to model the target and contexts. Using such a design, the model can effectively learn the representations of targets and contexts, which can jointly enhance the overall performance of target-based sentiment
Table 2: Accuracy comparison between the proposed IA-LSTM model performance for the TBSA task and the baseline results based on SVM along with n-gram features and approaches from related work.

<table>
<thead>
<tr>
<th>Model</th>
<th>Hotel R</th>
<th>Book R</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>74.26</td>
<td>71.50</td>
</tr>
<tr>
<td>INSIGHT-1</td>
<td>82.7</td>
<td>-</td>
</tr>
<tr>
<td>AB-LSTM-PC</td>
<td>82.60</td>
<td>-</td>
</tr>
<tr>
<td>IAN</td>
<td>81.90</td>
<td>78.96</td>
</tr>
<tr>
<td>IA-LSTM</td>
<td>83.10</td>
<td>80.82</td>
</tr>
<tr>
<td>Baseline</td>
<td>76.40</td>
<td>29.70</td>
</tr>
</tbody>
</table>

Table 3: Accuracy for each class in both Hotel reviews and Book reviews datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Positive</th>
<th>Negative</th>
<th>Neutral</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hotel R</td>
<td>89.86</td>
<td>83.29</td>
<td>76.14</td>
</tr>
<tr>
<td>Book R</td>
<td>86.42</td>
<td>87.93</td>
<td>65.82</td>
</tr>
</tbody>
</table>

different approaches show that the more attention that is paid to targets, the higher the accuracy.

As we can notice for Table 1, the class distribution is imbalanced in both Hotel review and Book review datasets. Therefore, we calculate the accuracy for each polarity class in Table 3. A lower accuracy obtained by class "Neutral" as it has the lowest amount of data and a higher accuracy obtained by class "Positive".

Lastly, we verified the effectiveness of using targets and interaction with the left and right contexts in modeling the attention mechanism and the results are displayed in Table 4. The first model (labelled No-interaction) completely ignored the interaction between the targets and contexts. This model uses three LSTM networks to learn the representations of the left context, target, and right context in their own local attentions without any interaction. In the second model (labelled Right-side interaction), we showed the impact of using the right-side of the interaction mechanism that enables the target to interact with the right contexts. This model uses two LSTM networks to learn the representations of the target and right context and interact with the right context only. Similarly, in the third model (labelled Left-side interaction) the target interacts with the left context only. The fourth model (labelled Full-model) uses the target interaction with both left and right contexts. As shown in Table 4, a lower performance was achieved by the No-interaction model. Using two-side interaction showed some improvement in the performance specially the left-side as Arabic is written from right to left which means most of the opinion words comes on the left-side after the target. The best performance was achieved by the Full-model, as we expected, where using the target interactions with the left and right contexts is fully considered, which enhances the overall model performance. Therefore, from this table, we can observe that the interaction of the target between the left and right contexts can contribute greatly in enhancing TBSA. The performance investigation of using target-attention only was well represented by the AB-LSTM-PC model in Table 2 which we avoid to repeat it in our comparison in Table 4.

6 Conclusion

In this paper, we proposed a deep learning-based approach to tackle Arabic TBSA. The proposed model “IA-LSTM” uses an interactive attention-based technique for the task. The main idea of the proposed IA-LSTM model is to use three attention networks to interactively model the target and contexts (left and right). It is based on previous work in English that uses only the right context but given the way Arabic is written from right to left, the addition of the left context provides important context information. The model can focus on the important parts in the sentence and identify the sentiment polarity. The proposed approach was evaluated on two different datasets: Arabic hotel reviews and book reviews. Experiments verify that the proposed approach outperforms the baselines and other approaches evaluated on the same datasets. Implementing the interactive attention-based model demonstrated that the model can learn effective features for targets and contexts (left and right) and enhance the performance of Arabic TBSA.
References


Hamed Al-Rubaeie, Renxi Qiu, and Dayou Li. 2016. Identifying mubasher software products through sentiment analysis of arabic tweets. In 2016 International Conference on Industrial Informatics and Computer Systems (CIICS), pages 1–6. IEEE.


Litescale: A Lightweight Tool for Best-worst Scaling Annotation

Valerio Basile
University of Turin
valerio.basile@unito.it

Christian Cagnazzo
University of Turin
christian.cagnazzo@edu.unito.it

Abstract

Best-worst Scaling (BWS) is a methodology for annotation based on comparing and ranking instances, rather than classifying or scoring individual instances. Studies have shown the efficacy of this methodology applied to NLP tasks in terms of a higher quality of the datasets produced by following it. In this system demonstration paper, we present LITESCALE, a free software library to create and manage BWS annotation tasks. LITESCALE computes the tuples to annotate, manages the users and the annotation process, and creates the final gold standard. The functionalities of LITESCALE can be accessed programmatically through a Python module, or via two alternative user interfaces, a textual console-based one and a graphical Web-based one. We further developed and deployed a fully online version of LITESCALE complete with multi-user support.

1 Introduction

Annotation is a cornerstone of Computational Linguistics and Natural Language Processing (NLP). Much of modern NLP is based on machine learning, and in particular on supervised machine learning. As a consequence, large amounts of manually annotated data are constantly in high demand, and the quality of the annotation directly influence the predictive capability of models trained on them.

The annotation of natural language resources comes in many forms of varying complexity and levels of abstraction. However, it is possible to categorize most of the approaches in three main families. Categorial annotation is perhaps the most common approach, whereas each instance of a dataset is associated with a label from a fixed set of options. Annotation can also consist of scalar values, that is, numeric values on a predetermined scale. Finally, ranking is a type of annotation where multiple instances are put in a certain order by the annotator, who therefore does not make judgments on instances in isolation but rather on groups of them.

Recent literature highlights the advantages of the ranking strategy, in terms of the quality of the annotation produced with it, in particular when dealing with subjective aspects of natural language (Yannakakis et al., 2018). The Best-Worst Scaling model (BWS) is a ranking-based annotation process developed by Louviere et al. (2015). BWS has been proved to be beneficial to the quality of the resulting gold standard data for subjective-related tasks such as emotion detection (Kiritchenko and Mohammad, 2017) and hate speech detection (Piletto et al., 2019). The recent work of De Bruyne, Luna and De Clercq, Orphée and Hoste, Veronique (2021) further proved the validity of BWS as a method of annotation for sentiment-related tasks, dominance in particular.

Several general-purpose annotation tools have been proposed in the literature. Among the most popular, WebAnno (Eckart de Castilho et al., 2016) and Brat (Stenetorp et al., 2012) both provide Web-based interfaces and the possibility to implement rich annotation schemas, including annotation at the word and span level, and links between annotations. A number of online services are also available to perform linguistic annotation, especially in a crowdsourced fashion, such as Amazon Mechanical Turk1, Appen2, or LightTag3. All these systems, however, allow the user to devise schemas that associate labels or scores to individual units of text, whether words, spans, sentences or others. In other words, they natively give the possibility of implementing categorial or ranking annotations. To our knowledge, there is no publicly available system

---

1https://www.mturk.com/
2https://appen.com/
3https://www.lighttag.io/
2 Best-worst Scaling Annotation

In Best-worst Scaling (BWS) annotation, the annotator is shown a tuple consisting of a fixed number of instances, and a phenomenon to annotate. The annotator is then asked to select the instance from the tuple which expresses the phenomenon to the maximum extent, and the one that expresses it to the least extent. For example, in a sentiment polarity annotation task, the annotator may be asked to select the sentences conveying the most positive and the most negative sentiment.

Starting from the set of instances to annotate, tuples must be created such that:

- The instances appear in tuples in a random order.
- Each instance appears in a predetermined number of tuples.
- The same pair of instances never appears in the same tuple more than once.

The size of the tuples and the number of tuples in which each instance appears are parameters to be determined at the time of the creation of the tuples. The ratio between the values of these two parameters determines the number of tuples with respect to the number of input instances. In particular, if the two parameters are equal, then the number of tuples will be approximately the same as the instances to annotate.

Since the number of tuples where any individual instance appears equals the size of the tuples (four items), the number of final questions for the annotators is roughly equivalent to the number of items to be annotated.

Once the annotation is complete, a score is computed for each instance as the difference between the number of times the instance was selected as “best” and the number of times it was selected as “worst”. The score can easily be normalized to fit a \([-1, 1]\) or \([0, 1]\) interval, depending on the needs of the downstream application.

Since the tuples are computed once, at the beginning of the annotation process, every annotator is presented the same set of instances. It is therefore possible to compute standard measures of inter-annotator agreement such as Fleiss’ Kappa or Krippendorf’s Alpha (Artstein and Poesio, 2008).

3 Litescale

In this section, we introduce LITESCALE, a free and open source software package that implements the Best-worst Scaling annotation framework. LITESCALE is composed of a core Python library implementing the core functions, and two alternative interfaces, a console-based textual one and a Web-based graphical one. The user can create annotation projects by providing a tab-separated text file, and obtain a similarly formatted file at the end of the annotation. A diagram depicting the information flow and the modules of LITESCALE is in Figure 1.

3.1 Core Functions

The core library of LITESCALE implements a series of functions to create and manage BWS annotation tasks, called projects. A project is created starting from a simple text file, where each line is a tab-separated pair of an instance identifier (which can be any string) and its textual content. Upon the creation of a new project, a JSON file is created representing all the information relative to the project.

Tuple creation. The function to create a new project takes five inputs: a name, the label for the phenomenon to annotate, the file containing the instances, and two parameters for the creation of the tuples from the set of input instances. Such parameters are the size of the tuples to create (tuple size) and the number of tuples in which a single instance will occur (instance replication). Given these two parameters, the system implements the three constraints described in Section 1 by leveraging Gauss’ modular arithmetic. More precisely, calling \( s \) the tuple size and \( r \) the replication factor, and \( n \) the
total number of instances, tuple are composed of instances with indexes in the form:

\[(xs^{j+1} + is^{j}) \mod n\]

\[0 \leq x < \lfloor n/s \rfloor\]
\[0 \leq i < s\]
\[0 \leq j < r\]

This formula ensures that the BWS conditions are met, by “wrapping up” the indexes in an appropriate way. For instance, with \(s = 4, r = 4,\) and \(n = 101,\) the first tuples are identified by the following indexes: \((1, 2, 3, 4), (5, 6, 7, 8), \ldots\) However, once the indexes exceed \(n,\) the modular arithmetic recombinates them: \((1, 5, 9, 13), \ldots, (81, 85, 89, 93), (97, 101, 4, 8),\) and so on. While this method ensures to fully include all the input instances into the tuples, due to the inner working of the modular arithmetic the formula works correctly under the condition that \(n\) (the number of instances) and \(s\) (the size of the tuples) are be co-primes, i.e., they need not have common divisors other than 1. As a workaround, the LITESCALE library automatically exclude a number up to \(s – 1\) of instances from the process, to prevent the formula to run into a corner case. Furthermore, for some combinations of parameters it is mathematically impossible to have all the instances appear in exactly \(r\) tuples. More precisely, there will be up to \(s – 1\) instances that occur \(r – 1\) times, i.e., a considerably small set of instances will lack exactly one annotation at the end of the process.

**BWS annotation.** The library creates a directory to store all the annotations for a project as a series of JSON files, one for each user participating to the annotation task. The `next tuple` function checks the current status of the annotation and returns the first non-annotated tuple to be presented to the user. The `annotate` function takes a pair of tuple indexes indicating the “best” and the “worst” and updates the annotation JSON correspondingly. The `progress` function returns the number of currently annotated tuples by a given user and the total number of tuples in the project, in order to compute the advancement status of the project as a percentage or progress bar.

**Gold standard.** The other key function in the LITESCALE core library is the one that computes the gold standard. This function collects all the annotations provided by the users for a specific project, and computes the gold standard according to the BWS methodology. For each instance, library counts the number of times it was judged “best” and the number of times it was judges “worst”, and computes the difference between these numbers. The result is then normalized using the minimum and maximum values of the differences across all instances, in order to return a score between 0 and 1, indicating the relevance of the phenomenon object of the annotation project in each instance. The gold standard dataset is finally written out to a text tab-separated file consisting of three columns, namely instance ID, text, and score.

### 3.2 Command-line Interface

The command-line interface (CLI) of LITESCALE is implemented with the [PyInquirer](https://github.com/CITGuru/PyInquirer) Python library. It makes use of the functions provided by the LITESCALE core library and exposes an interactive, menu-based interface to easily navigate through its functionalities.

![Figure 2: LITESCALE command-line interface: login and start menu.](image1)

Upon starting, the CLI asks for a username to log in, and keeps memory of the last user that has logged in previous sessions. No password is requested, since the application is intended to be for single user interaction in individual installations. After logging in, a menu with a few options is proposed to the user, as shown in Figure 2.

![Figure 3: LITESCALE Command-line interface: creation of a new annotation project.](image2)

- **Start/continue annotation**: lists the available annotation projects, and start the annotation of the selected project.
- **Generate gold standard**: lists the available annotation projects, and creates the gold stan-
The creation of a new annotation project involves a small set of questions for the user. These include a name for the project, a label for the phenomenon to annotate, values for the two parameters for the creation of the tuples (tuple size and instance replication), and the path to a tab-separated file from which the instances will be read.

For the BWS annotation, the CLI shows two questions in sequence to the user, in the form “which is the MOST label?” and “which is the LEAST label?”, followed by the list of the instances in a tuple, where label is the label of the phenomenon for the selected project. An example of the annotation interface is shown in Figure 4. At any moment, the user can select PROGRESS to check how many instances are left to complete the task (Figure 5), or EXIT to return to the main menu.
of LITESCALE is shown, initially prompting for a username, exactly as its CLI counterpart. Once the user has logged in, the main menu is displayed as in Figure 6.

The Web-based interface provides the same functionalities as the CLI, and the annotation workflow is identical (Figure 7). Furthermore, the two interfaces are by default part of the same installation of LITESCALE and make use of the same core library. Therefore, an annotation task can start with one interface and switch seamlessly to the other and back at any time. The annotations created with the two interfaces are exactly in the same format, and can therefore be merged without any extra work.

4 Multi-user Online Platform

Despite being a standalone application, LITESCALE is natively multi-user, because the annotations produced by different users are represented as different JSON files. These files can reside on the same machine, or different machines if multiple installation of LITESCALE took place. In the latter case, it is sufficient to copy the annotation files of a project from one installation to another to generate a gold standard comprising all the annotations. This process, however, can be tedious and error-prone. Moreover, the version of LITESCALE presented so far needs to be executed as a series of Python scripts, therefore requiring at least a minimal amount of skill in such technology from the annotators. To overcome these issues, and provide an even easier and more accessible user experience, we developed a fully online, multi-user version of LITESCALE, described in this section.

The online version of LITESCALE offers the same functionalities as the standalone version described in the previous section, with a user interface very similar to the Web-based interface in Section 3.3. The front-end was however implemented from scratch, while retaining the core library for the basic functions, although with some modifications. LITESCALE online is a Web application implemented with the Flask Python framework. More precisely, the software is designed in a modular way. A RESTful HTTP API is provided in order to expose the core library functionalities, and a Web application provides the user interface by connecting to the API. Figure 8 shows the modular architecture of this version of LITESCALE.

The RESTful API is intended as a layer of abstraction to facilitate the future development of tools that access the core library functions remotely, or application that provide alternative interfaces, such as mobile applications. The RESTful API is implemented with the Flask-RESTful extension of the Flask framework. The HTTP verbs and their semantics are summarized in Table 1.

In order to make LITESCALE online ready to scale up the number of users and projects, the core library has been modified. In particular, the inner representation and persistence of instances, projects, tuples, and annotations, does not rely on JSON files but rather on a SQL database.

The workflow in the online version of LITESCALE is substantially the same as the standalone version, including the interface to create new projects (Figure 9) and the annotation interface itself. However, additional functionalities were implemented to account for the online multi-user envi-

---

6https://flask.palletsprojects.com

7https://flask-restful.readthedocs.io
Table 1: The LITESCALE RESTful API.

<table>
<thead>
<tr>
<th>Method</th>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/users</td>
<td>Creates a user</td>
</tr>
<tr>
<td>DELETE</td>
<td>/users</td>
<td>Deletes a user</td>
</tr>
<tr>
<td>GET</td>
<td>/projectList</td>
<td>Retrieve the list of projects of a user</td>
</tr>
<tr>
<td>GET</td>
<td>/projects</td>
<td>Retrieves the properties of a project</td>
</tr>
<tr>
<td>POST</td>
<td>/projects</td>
<td>Creates a project</td>
</tr>
<tr>
<td>DELETE</td>
<td>/projects</td>
<td>Deletes a project</td>
</tr>
<tr>
<td>GET</td>
<td>/tuples</td>
<td>Retrieves the tuples of a project</td>
</tr>
<tr>
<td>POST</td>
<td>/annotations</td>
<td>Creates a new annotation</td>
</tr>
<tr>
<td>GET</td>
<td>/gold</td>
<td>Generates the gold standard</td>
</tr>
<tr>
<td>GET</td>
<td>/progress</td>
<td>Returns the current progress of an annotation task</td>
</tr>
<tr>
<td>POST</td>
<td>/authorizations</td>
<td>Adds a user to a project</td>
</tr>
<tr>
<td>DELETE</td>
<td>/authorizations</td>
<td>Removes a user from a project</td>
</tr>
</tbody>
</table>

environment. The users can sign up with a valid email address and set up proper log in credentials after receiving a confirmation by email. The user authentication features are managed by the Werkzeug\(^8\) Python library, including the storage of encrypted passwords in the database. Moreover, the association of users to projects is also managed through the Web application. By default, when a project is created, the user who created it assumes the role of owner of that project. The owner of a project can invite other users to join their projects, by indicating a valid email address (Figure 10). The recipients of such invitation will receive an automated email notification, inviting them to sign up to LITESCALE if they are not already registered.

5 Evaluation

We conducted a pilot test in order to evaluate the efficacy of Litescalse in supporting the Best-worst Scaling annotation of datasets for NLP tasks. We extracted a sample of short hotel reviews in English from the list of 1,000 hotels and their reviews provided by Datafiniti’s Business Database\(^9\). We selected the 20 shortest reviews for each of the five ratings (1 to 5 stars) for a total of 100 instances, shuffled then randomly and created an annotation task in Litescalse. Three annotators were asked to perform the annotation and record the times spent annotating each 10 annotations (pairs of best/worst judgments). We computed the Pearson correlation between the scores resulting from Litescale and the original ratings associated to the reviews, obtaining a score of 0.77, indicating strong correlation with the ground truth. The agreement between the annotators is relatively high, considering the subjectivity of the sentiment polarity annotation task. Interestingly, the three annotators agreed more on the worst judgments (two annotators agreed 91% of the times, all three agreed 66% of the times) than on the best ones (two annotators agreed 88% of the times, all three agreed 58% of the times).

The annotation took about 43 minutes on average. However, the average time spent annotating tends to decrease consistently as the annotation task progresses, as shown in Figure 11. This is not surprising, considering the nature of the BWS annotation, where the same instance is shown multiple times, and therefore the annotator gains familiarity with the instances over time.

6 Conclusion and Future Work

We introduced LITESCALE, a flexible software tool to create and manage linguistic annotation tasks based on the Best-worst scaling methodology. LITESCALE is easy to use, free and open source\(^10\), and ships with different user interfaces. Moreover, a multi-user online version of LITESCALE is also presented\(^11\), which runs entirely in a Web browser.

---

\(^8\)https://werkzeug.palletsprojects.com
\(^10\)The source code of LITESCALE is available on the Github repository https://github.com/valeriobasile/litescale
\(^11\)LITESCALE online is available at http://lite-env.eba-jhijbmtj.eu-west-3.elasticbeanstalk.com/home
and provides all the functionalities of the original software.

While the BWS methodology has been thoroughly tested in the literature, this particular tool has not been systematically tested for its usability (but it is being used at the moment for the creation of several language resources). Although all the main features are implemented, there is always room to add extra functionalities and improving the existing ones.

A short video describing the main functions of LITESCALE is available on YouTube: https://youtu.be/SozWDMH2ah0

Acknowledgments

This work is partially funded by the project “Be Positive!” (under the 2019 “Google.org Impact Challenge on Safety” call).

References


De Bruyne, Luna and De Clercq, Orphée and Hoste, Veronique. 2021. Annotating affective dimensions in user-generated content: comparing the reliability of best-worst scaling, pairwise comparison and rating scales for annotating valence, arousal and dominance. *LANGUAGE RESOURCES AND EVALUATION*.


Probabilistic Ensembles of Zero- and Few-Shot Learning Models for Emotion Classification

Angelo Basile
Symanto Research
Nuremberg, Germany
angelo.basile@symanto.com

Guillermo Pérez-Torró
Symanto Research
Valencia, Spain
guillermo.perez@symanto.com

Marc Franco-Salvador
Symanto Research
Valencia, Spain
marc.franco@symanto.com

Abstract

Emotion Classification is the task of automatically associating a text with a human emotion. State-of-the-art models are usually learned using annotated corpora or rely on hand-crafted affective lexicons. We present an emotion classification model that does not require a large annotated corpus to be competitive. We experiment with pretrained language models in both a zero-shot and few-shot configuration. We build several of such models and consider them as biased, noisy annotators, whose individual performance is poor. We aggregate the predictions of these models using a Bayesian method originally developed for modelling crowdsourced annotations. Next, we show that the resulting system performs better than the strongest individual model. Finally, we show that when trained on few labelled data, our systems outperform fully-supervised models.

1 Introduction

A large part of Natural Language Processing (NLP) research is focused on building technology to automatically extract information from large collections of texts. However, text contains often not just mere factual information, but also opinions, attitudes and emotions. Applications of emotion-aware NLP models range from established tasks such as analysis of product reviews (Blitzer et al., 2007) and development of “emotional” chatbots (Chatterjee et al., 2019) to less obvious tasks such as the analysis of developer experience on Stack Overflow (Novielli et al., 2018), author profiling (Rangel and Rosso, 2016) and the prediction of mental health disorders (Uban et al., 2021).

In this work, we focus on the fine-grained emotion classification task (Strapparava and Mihalcea, 2007): given a document, an emotion label must be provided. For example, the sentence The angry wolf ate the happy boy could be associated with fear or sadness if the emotion is being modelled from the reader’s perspective; alternatively, it could be associated with anger or joy considering the text’s perspective. In the emotion classification literature, the targeted emotion perspective is rarely made explicit (Bostan et al., 2020), which — in addition to the subjectivity involved in the annotation task — makes it difficult to obtain large amounts of high quality data (Bobicev and Sokolova, 2017; Troiano et al., 2021). Furthermore, with few exceptions (Mohammad et al., 2018; Lamprinidis et al., 2021), most of the research has been conducted on English corpora: most of the other languages can be considered low-resourced with respect to affective corpora.

In this work, we aim to minimize the amount of annotated data needed to obtain competitive performance in the task of emotion classification. Several emotion theories exist, which differ on the emotion inventory and representation type (categorical vs. continuous): in this work we focus on the categorical paradigm, using the mix of different inventories available from the Unify Emotion dataset (Bostan and Klinger, 2018).

We describe the use of pretrained language models (PLMs) for emotion classification in both few-shot and zero-shot scenarios (Section 4). Few-shot models are supposed to solve a task using only few annotated instances; zero-shot models are supposed to use none. These models have been shown to perform well in different tasks (Yin et al., 2019; Schick and Schütze, 2021b; Wang et al., 2021). However, in a real unsupervised scenario (i.e., without...
an evaluation split), their performance is by definition unknown. To mitigate the risks involved in deploying such models, we experiment with a probabilistic ensemble that combines the individual – and potentially biased and noisy – outputs of those models. We use the Multi-Annnotator Competence Estimation model (Hovy et al., 2013), a Bayesian method designed to deal with noisy crowdsourced annotations (Section 5). Experimental results (Section 6) show that our ensemble performs better than the strongest individual model. In addition, we show that just fine tuning with few labeled data, our system outperforms fully-supervised models.

2 Related Work

Attempts to minimize the amount of hand-labelled data required to train emotion-aware NLP models have mostly focused on using distant supervision (Go, 2009) to collect large amounts of silver labels for training models in a supervised fashion: emoji (Felbo et al., 2017), emoji description (Eisner et al., 2016), and hashtags (Mohammad, 2012) have been shown to be good proxies for emotion classification. The idea of using label templates for unsupervised classification can be traced back at least to Hearst patterns (Hearst, 1992). Within the neural paradigm, Cloze (Taylor, 1953) label templates are used by Schick and Schütze (2021a), who obtain strong results on few-shot classification. Yin et al. (2019) and Wang et al. (2021) use templates to generate synthetic data for framing text-classification as entailment. An alternative neural approach to unsupervised classification embeds both the input sequence of text and the set of possible label names in the same semantic space and selects the one which maximizes a defined similarity metric (Gabrilovich and Markovitch, 2007). Pushp and Srivastava (2017) concatenates both input and label embeddings and classify their relatedness. A recent survey of template-based or “prompt-based” learning can be found in Liu et al. (2021). Perhaps the closest work to ours is Yin et al. (2019), who evaluate zero-shot text classification on the Unify Emotion dataset (Bostan and Klinger, 2018).1 Our main contribution with respect to Yin et al. (2019) is the suggestion of a principled way for a) aggregating multiple predictions without having any access to a model’s performance and b) inferring the most probable emotion label given multiple models.

An overview of Bayesian models of annotation can be found in Paun et al. (2018). The idea of using a generative model to infer a latent label from multiple signals has recently been presented in a unified framework by Ratner et al. (2016). An alternative to Bayesian models for aggregating predictions can be found in Poerner and Schütze (2019), who apply Generalized Canonical Correlation Analysis to build an ensemble of unsupervised BERT models for Duplicate Question Detection in a low-resource scenario.

3 Data

<table>
<thead>
<tr>
<th></th>
<th>train</th>
<th>validation</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>anger</td>
<td>5147</td>
<td>1714</td>
<td>1717</td>
</tr>
<tr>
<td>antipation</td>
<td>191</td>
<td>64</td>
<td>63</td>
</tr>
<tr>
<td>confusion</td>
<td>77</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>disgust</td>
<td>2701</td>
<td>900</td>
<td>899</td>
</tr>
<tr>
<td>fear</td>
<td>9592</td>
<td>3196</td>
<td>3199</td>
</tr>
<tr>
<td>guilt</td>
<td>656</td>
<td>218</td>
<td>219</td>
</tr>
<tr>
<td>joy</td>
<td>22338</td>
<td>7448</td>
<td>7446</td>
</tr>
<tr>
<td>love</td>
<td>2292</td>
<td>764</td>
<td>764</td>
</tr>
<tr>
<td>noemo</td>
<td>62692</td>
<td>20897</td>
<td>20898</td>
</tr>
<tr>
<td>sadness</td>
<td>9185</td>
<td>3061</td>
<td>3061</td>
</tr>
<tr>
<td>shame</td>
<td>658</td>
<td>219</td>
<td>219</td>
</tr>
<tr>
<td>surprise</td>
<td>5392</td>
<td>1795</td>
<td>1798</td>
</tr>
<tr>
<td>trust</td>
<td>485</td>
<td>162</td>
<td>161</td>
</tr>
</tbody>
</table>

Table 1: Overview of the dataset.

For all our experiments we use a section of the Unify Emotion dataset (Bostan and Klinger, 2018) which aggregates several annotated corpora in a common format. Specifically, we use the following datasets: Grounded-Emotions (Liu et al., 2007), CrowdFlower (Crowdflower), DailyDialog (Li et al., 2017), TEC (Mohammad, 2012), Electoral-Tweets (Mohammad et al., 2015), ISEAR (Scherer and Wallbott., 2017), Emotion-Stimulus (Ghazi et al., 2015), Tales-Emotion (Alm et al., 2005; Alm and Sproat, 2005; Alm, 2008), and EmoInt (Mohammad et al., 2017). We aggregate all the corpora and then sample from each class 60% of the data for the train split, and 20% for the Emotion dataset, the actual instances and label set used are different.

1We don’t compare our results directly to Yin et al. (2019): even though both datasets stem from the Unify
development and the test split, respectively. The annotation quality, domain, annotation procedure (manual vs. semi-automatic), and topic differ among the various datasets. We refer to (Bostan and Klinger, 2018) for additional details about the specific datasets.

Table 1 highlights the label distribution in the dataset. As it is the case for most available emotion corpora, the labels are heavily unbalanced.

4 Entailment as Zero-Shot and Few-Shot Learning

Given two sentences, a premise and a hypothesis, they can be related by an entailment, contradiction or neutral relation. The task of Natural Language Inference (NLI) (Dagan et al., 2005) aims at predicting such relations. Recently, the creation of large NLI datasets (Bowman et al., 2015; Williams et al., 2018; Thorne et al., 2018; Conneau et al., 2018) has allowed deep learning methods to achieve state-of-the-art performance on the NLI task, outperforming logic-based approaches. The high performance of BERT-like models (Devlin et al., 2019) on NLI tasks can be exploited to successfully tackle general classification tasks by recasting them as entailment problems: pre-trained language models can be finetuned on NLI datasets and these finetuned models can be then re-purposed to attack different problems (Wang et al., 2021). For modelling the emotion classification problem, we follow Yin et al. (2019) and given a text to classify (the hypothesis), we build pseudo-sentences to serve as premises, one for each target label. For instance, the input sentence “John said he loved the pizza” can be classified as JOY, if an NLI models predicts that it entails the artificial sentence “This person expressed a feeling of pleasure”. We can substitute pleasure with other emotion-expressing words and map them to specific target labels (e.g., pleasure to JOY, sad to SADNESS, etc.) to build a system for zero-shot, multi-label emotion classification.

Following Yin et al. (2019), in this work we explore two options to formulate the hypotheses: based on the label’s name and on the label’s WordNet (Fellbaum, 1998) definition. We show the details about our hypotheses for emotion classification in Table 2.

We experiment with six different pretrained NLI models that differ in terms of the underlying pretrained language model (BART (Lewis et al., 2020), RoBERTa (Liu et al., 2019) and XLM-RoBERTa (Conneau et al., 2020)) and NLI dataset used for training (Multigenre NLI (MNLI) (Williams et al., 2018), Adversarial NLI (ANLI) (Nie et al., 2020) and XNLI (Conneau et al., 2018)). In Table 3.B of Section 6 we include the details about our models. The zero-shot setup motivates the usage of a variety of pretrained NLI models: given that in a true zero-shot scenario no development dataset is available, assessing how different NLI training data and pretrained language models impact the performance is of crucial importance.

We conduct the few-shot learning experiments by fine-tuning a pretrained entailment model. To build the training data, we fill the templates used in the zero-shot setup with the gold labels and training follows the standard sentence pair classification task used to train the original entailment model.

5 A Probabilistic Ensemble

In a true zero-shot classification scenario, no development set is available and therefore a method for estimating the performance of the model on the specific input data is required. In this work, we propose to use several different models and to infer the best possible answer using a probabilistic model. Such a model has two advantages over a simple majority voting strategy: first, it has been shown to outperform majority voting (Snow et al., 2008); second, it provides a confidence value for each instance and estimates the models’ accuracy.

To aggregate the predictions from the different unsupervised models, we use the Multi-Annotator Competence Estimation (MACE) model (Hovy et al., 2013). This model has been originally developed to analyse crowdsourced annotations for both identifying unreliable annotators and retrieving the true labels. Figure 1 shows the plate diagram of the model and we refer to the original publication for further details. Algorithm 1 describes the generative process.

We generalize the notion of annotator to also

---

2 We downloaded the models from https://huggingface.co.
Table 2: Formulation of label as hypotheses for entailment. All our hypotheses start with "This person (...)".

<table>
<thead>
<tr>
<th>Label</th>
<th>Label-based hypothesis</th>
<th>WordNet-based hypothesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>anger</td>
<td>(...) feels angry</td>
<td>(...) expresses a strong feeling of annoyance, displeasure, or hostility</td>
</tr>
<tr>
<td>(...)</td>
<td>(...) expresses a strong feeling of annoyance, displeasure, or hostility</td>
<td></td>
</tr>
<tr>
<td>(...)</td>
<td>(...) expresses an unpleasant emotion caused by the belief that someone or something</td>
<td></td>
</tr>
<tr>
<td>(...)</td>
<td>(...) expresses a feeling of having done wrong or failed in an obligation</td>
<td></td>
</tr>
<tr>
<td>(...)</td>
<td>(...) expresses a feeling of revulsion or strong disapproval aroused by something</td>
<td></td>
</tr>
<tr>
<td>(...)</td>
<td>(...) expresses an unpleasant emotion caused by the belief that someone or something</td>
<td></td>
</tr>
<tr>
<td>(...)</td>
<td>(...) expresses an unpleasant emotion caused by the belief that someone or something</td>
<td></td>
</tr>
<tr>
<td>(...)</td>
<td>(...) expresses a feeling of great pleasure and happiness</td>
<td></td>
</tr>
<tr>
<td>(...)</td>
<td>(...) expresses a strong interest and pleasure in something</td>
<td></td>
</tr>
<tr>
<td>(...)</td>
<td>(...) expresses emotions experienced when not in a state of well-being</td>
<td></td>
</tr>
<tr>
<td>(...)</td>
<td>(...) expresses a painful feeling of humiliation or distress caused by the consciousness</td>
<td></td>
</tr>
<tr>
<td>(...)</td>
<td>(...) expresses a feeling of mild astonishment or shock caused by something unexpected</td>
<td></td>
</tr>
<tr>
<td>(...)</td>
<td>(...) expresses a feeling of mild astonishment or shock caused by something unexpected</td>
<td></td>
</tr>
<tr>
<td>(...)</td>
<td>(...) expresses a strong belief in the reliability, truth, or ability of someone or something</td>
<td></td>
</tr>
</tbody>
</table>

Algorithm 1: Generative process of the MACE model.

\[
\text{for item } i \in I \text{ do} \\
\quad \text{draw } G_i \sim \text{Uniform}; \\
\quad \text{for annotator } n \in N \text{ do} \\
\quad \quad \text{draw } B_{i,n} \sim \text{Bernoulli} \left(1 - \theta_j\right); \\
\quad \quad \text{if } B_{i,n} == 0 \text{ then } \\
\quad \quad \quad \text{ } y_{i,n} = G_i; \\
\quad \quad \text{else} \\
\quad \quad \quad \left[ y_{i,n} \sim \text{Multinomial} \left(\xi_j\right) \right] \\
\]

Figure 1: The MACE model. Given \( I \) instances and \( N \) annotators, the observed label \( y_{i,n} \) is dependent on the gold label \( G_i \) and \( B_{i,n} \), which models the behaviour of annotator \( n \) on instance \( i \). The model parameters \( \theta \) and \( \xi \) are left out.

6 Evaluation

We evaluate the unsupervised models in a zero-shot configuration against two supervised baselines. We then experiment with adding increasing amounts of supervision to both the baselines and the entailment model in a few-shot setting. We conduct all the evaluation using the standard classification metrics: precision, recall, and macro-averaged f1-score.

Baselines As upper-baselines for our experiments, we train two supervised models on all the available training data: we train both a neural network based on RoBERTa-base (Liu et al., 2019) and a linear SVM using character n-grams as representations (henceforth referred to as Char-SVM).  

Entailment Models We assemble 12 different unsupervised zero-shot classification models finetuned on the NLI task. The models differ in terms of the pretrained language model, label template and NLI finetuning data. Yin et al. (2019) a) explore two different evaluation scenarios: label-partially-unseen and label-

----
3The original Java implementation can be found at https://github.com/dirkhovy/MACE

4We use a custom implementation of a RoBERTa and use the pretrained model from https://huggingface.co/models. For the SVM model, we use the LinearSVM implementation contained in scikit-learn (Pedregosa et al., 2011).
fully-unseen. In the partially-unseen setup, a model is trained on a subset of the label set and then evaluated on the full dataset; in the fully-unseen setup, no labelled data is shown to the model. In this work we do not take into account the label-partially-unseen because, as stated by Yin et al. (2019), this is a restrictive definition of the zero-shot paradigm, unlike the label-fully-unseen scenario. For the few-shot evaluation, we only train the best-performing model (distilbart-mnli-12-1). Information about the used hyperparameters can be consulted in A.

**Probabilistic Ensemble** We train a MACE model using Variational-Bayes on the predictions of the 12 zero-shot entailment models. We train the model for 100 iterations and 50 restarts; we use the default values (0.5) for the $\alpha$ and $\beta$ parameters.

**Results** As reported in Table 3, emotion classification is a challenging task even for fully-supervised models trained on large annotated datasets. Interestingly, RoBERTa-base, a large neural model, outperforms the CharSVM model only by few points. The battery of zero-shot models shows a large variation in terms of performance, ranging from 8.01% f1-score for XLM-RoBERTa-large with WordNet-based hypothesis to 20.12% for distilbart with label names. The performance of that distilled entailment model is remarkable, considering that bart-large uses twice the number of parameters of its distilled version. On average, name-based templates outperform WordNet-based ones. Aggregating the predictions of the zero-shot models using MACE leads to a much higher f1-score when compared to majority voting. The MACE-based ensemble outperforms the strongest zero-shot model in terms of f1-score by a small but statistically significant margin (+0.84% f1-score). However, in a true zero-shot scenario, where no evaluation

Table 3: Overview of the evaluation results. Scores are macro-averaged. L: embedded label name for hypothesis representation; W: WordNet definition for hypothesis representation. [ ]: distilled model. Rows in A: fully supervised. Rows in B: zero-shot. Rows in C: aggregations. Rows in D: few-shot learning, each row denotes the number of training instances; the results are averaged over three runs and the standard deviation is shown in subscript. Statistically significant results according to a $\chi^2$ test, per sub-table, are highlighted in bold. Significant results between Zero-shot and the aggregations (B and C sub-tables), are highlighted with *.
set is available, simply discarding the predictions from weak, unfit models and selecting the best ones available, is crucial for deploying zero-shot models in a production environment. Our results show that for emotion classification, the model-based aggregation can not only automatically select the best available model, but also improve its performance.

When few annotated instances are available, our results show that entailment models perform notably better than supervised models: Figure 2 shows that a finetuned entailment model outperforms by a large margin not only a linear baseline model using shallow features but also a strong neural LM-based model. The RoBERTa-base model is outperformed by the entailment model by a large margin (+6.98% f1-score) when trained on the full dataset. This highlights a key advantage of few-shot learning for under-resourced scenarios.

Given the diverse nature of the data that compose the Unify Emotion dataset, we evaluate four different models on the individual datasets contained in Unify Emotion: Table 4 highlights the results. As shown already in Bostan and Klinger (2018), some datasets are easier to model than others. CrowdFlower and DailyDialog are relatively noisy datasets and all the models struggle on them. Datasets containing noisy text written in non-canonical language (e.g., Electoral-Tweets, Grounded-Emotions), seem to challenge the zero-shot models more than corpora like ISEAR and Tales-Emotion which contains more standard text. ISEAR’s annotation format is particularly close to the pseudo-sentences that we used (i.e., “This person feels [...]”), which can explain the high performance achieved by the zero-shot model.

7 Conclusions

In this work we presented an emotion classification model that does not require large annotated data to be competitive on the Unify Emotion dataset. We experimented with pretrained language models in both the zero-shot and few-shot settings. We aggregated the predictions of these models using MACE, a Bayesian method developed for modelling noisy, crowdsourced annotations. Experimental results showed that the resulting system performs better than the strongest individual zero-shot model. When evaluated on a diverse dataset, our zero- and few-shot models behave in a comparable way to fully-supervised models, without requiring the same amount of annotated data. Noisy text seems to challenge the NLI models trained on canonical text, while zero-shot models perform well when the annotation scheme matches the pseudo-sentences used for building the synthetic data: this suggests that different domains might need different templates that take into account elements like vocabulary or stylistic variation. Finally, we showed that when the MACE and the few-shot systems are trained...
on few labelled data, they outperform fully-supervised models.

In future works we will further explore how to apply zero and few-shot learning for text classification tasks, and how to better aggregate the outputs of different models in an unsupervised manner.

References


Crowdflower. The emotion in text, published by crowdflower.


A. Go. 2009. Sentiment classification using distant supervision.


Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural


A Appendix

A.1 Hyperparameters

Training the Entailment model in a few-shot scenario was carried out using a single NVIDIA GeForce GTX 1080 Ti, which allows to allocate up to 11GB of RAM. This has constraint some of the hyperparameters we have chosen to try.

Aside from that, we have also followed the recommendations shared by Wang et al. (2021) because our experiments with few-shot learning were very similar to theirs.

**Batch size and Maximum Length** A batch size of 8 samples was used. The maximum length was adapted to the maximum number of tokens seen when encoding the whole Unified dataset with the corresponding tokenizer of the chosen model (distilbart-mnli-12-1). When computing this, it was also taken into account the fact that, for the entailment approach, the label description or hypothesis is encoded as an additional input to the model. The final selected value was 286 tokens.

**Learning Rate** Typical learning rate values recommended for fine tuning an Adam optimizer are: 5e-5, 3e-5, 2e-5 (Devlin et al., 2019). Following the implementation of Wang et al. (2021), we used a constant and smaller value of 1e-5.

**Epochs** As a practical consideration we decided to train just 1 epoch because we observed that training on more steps reduced the overall performance.

**Number of trials** In order to avoid instability among reported results, mainly caused by the small number of samples used in few-shot experiments (Wang et al., 2021; Gao et al., 2021), the metrics measuring the performance of the model are averaged among 3 different runs that are trained over its corresponding randomly sampled training sets from the whole Unified dataset.
Cross-Lingual Wolastoqey-English Definition Modelling

Diego Bear  
Faculty of Computer Science  
University of New Brunswick  
diego.bear@unb.ca

Paul Cook  
Faculty of Computer Science  
University of New Brunswick  
paul.cook@unb.ca

Abstract

Definition modelling is the task of automatically generating a dictionary-style definition given a target word. In this paper, we consider cross-lingual definition generation. Specifically, we generate English definitions for Wolastoqey (Malecite-Passamaquoddy) words. Wolastoqey is an endangered, low-resource polysynthetic language. We hypothesize that sub-word representations based on byte pair encoding (Sennrich et al., 2016) can be leveraged to represent morphologically-complex Wolastoqey words and overcome the challenge of not having large corpora available for training. Our experimental results demonstrate that this approach outperforms baseline methods in terms of BLEU score.

1 Introduction

Definition modelling, introduced by Noraset et al. (2017), is the task of automatically generating a dictionary-style definition for a given target word. Definition modelling can provide more-transparent, human-interpretable representations of the information in embeddings. Definition modelling could also potentially be applied to automate, or semi-automate, the constructing or updating of dictionaries, for example, by generating draft definitions for newly-emerged words that are not yet listed. Although there has been a range of work on definition modelling (e.g., Ni and Wang, 2017; Gadetsky et al., 2018; Chang and Chen, 2019) the focus has been on monolingual definition modelling, with the target word and generated definition being in the same language.

Malecite-Passamaquoddy (also Maliseet-Passamaquoddy, Passamaquoddy-Maliseet) is an Eastern Algonquian language spoken in regions of what is now New Brunswick and Quebec, Canada, and Maine, United States. Malecrite and Passamaquoddy are dialects of this language. However, Malecite is a Mi’kmaq exonym, with Wolastoqey being the term this speech community uses to refer to their language. We therefore use the term Wolastoqey throughout this paper.

Wolastoqey is an endangered language, with roughly 300 remaining first language speakers in Canada (Statistics Canada, 2017). Moreover, children are typically not learning the language proficiently. Wolastoqey is also a low-resource language, with no large corpora or annotated datasets available for training natural language processing (NLP) systems. However, the Passamaquoddy-Maliseet Dictionary (Francis and Leavitt, 2008) is available online through the Passamaquoddy-Maliseet Language Portal. This dictionary includes roughly 19k entries with Wolastoqey headwords and English definitions. Many entries also include parallel Wolastoqey-English example sentences. There has been very little prior computational work on Wolastoqey, with Farber (2015) presenting a preliminary finite-state model of nouns.

Wolastoqey, like other Algonquian languages, is polysynthetic. Verbs in particular have rich morphological structure, and often include several roots (Leavitt, 1996). Consider the example gloss below for paskoloqessu:

<table>
<thead>
<tr>
<th>pask-oloq-ess-u</th>
</tr>
</thead>
<tbody>
<tr>
<td>breaking-ice-move.quickly-s/he</td>
</tr>
</tbody>
</table>

‘She or he moves quickly across ice as it cracks’

The root *oloq* can be seen in various other words, such as ’*ketoloqtehmon* ‘s/he chips it out of ice’, *sahsoloqe* ‘it is slippery, is icy’, and *supoloqe* ‘there is smooth ice (on lake, etc.)’. There is, however, ambiguity in that the character sequence *oloq* does not always correspond to this morpheme. For example, in *oloqapeku* ‘s/he crawls in that direction’ *oloq* has the meaning of ‘in that direction’.

All examples are taken from the Passamaquoddy-Maliseet Language Portal.

In this paper we propose a model for cross-lingual Wolastoqey-English definition modelling. We hypothesize that sub-word representations of Wolastoqey words based on byte pair encoding (BPE) tokenization (Sennrich et al., 2016) can be leveraged to generate English definitions. We propose a sequence-to-sequence model (Sutskever et al., 2014) in which the encoder operates over Wolastoqey words segmented via BPE, and the decoder generates English definitions. We show that our proposed model is able to outperform baseline systems in terms of BLEU score.

Wolastoqey speakers regularly create new words by creatively combining roots (Leavitt, 1996). As such, not all words can be expected to be included in a dictionary. Cross-lingual Wolastoqey-English definition modelling could therefore be helpful for Wolastoqey learners.

2 Related Work

The task of definition modelling is to learn to generate a dictionary-style definition for a given input word. This task was initially described by Noraset et al. (2017), who focused on generating English definitions for English words. Noraset et al. proposed a word-to-sequence neural language model, composed of a two-layer LSTM and a parallel CNN, to generate a definition given an initial input word and its embedding. This language model-based approach generates definitions by iteratively predicting the next occurring word given some prior history. In this model, different inputs were given to each of the sub-components. The LSTM component was initially given the embedding for the word being defined, but also considered the embeddings of its previous output in the form of context at a given timestep. The CNN sub-network, on the other hand, was used to extract character-level information about the word and, as such, was given the characters of the word being defined. The CNN was included because the word-level LSTM has no knowledge of sub-word information. Manual analysis of the proposed system’s performance considered seven types of errors that were observed to occur in the generated definitions. These include redundancy, self-reference, wrong part-of-speech, under-specification, opposite definition, close semantic errors, and incorrect definition generation. Out of all these errors, incorrect definition generation was observed to be the most common. This paper further found that high-quality word embeddings were crucial for definition modelling to be successful.

One challenge for our work is that we do not have a large corpus available from which to learn high-quality Wolastoqey word embeddings. We propose to use BPE segmentation to overcome this. Specifically, we hypothesize that sub-word representations based on BPE can be leveraged to represent morphologically-complex Wolastoqey words without requiring a large corpus to be available for training word-level embeddings.

The approach of Noraset et al. (2017) is context-agnostic; i.e., the model generates a definition for a target word without any specific context of usage for the target. Other context-agnostic approaches to definition modelling include Yang et al. (2020) who incorporate knowledge of Chinese sememes (minimum semantic units) for Chinese definition modelling, and Balachandran et al. (2018) who propose a domain-specific definition generation model for the software domain. In line with these previous studies, we also propose a context-agnostic approach.

Contrasting with context-agnostic approaches, context-aware approaches to definition modelling have also been considered (e.g., Ni and Wang, 2017; Gadetsky et al., 2018; Mickus et al., 2019). In these approaches a definition is generated for a target word used in a specific context. Some context-aware methods have used a sequence-to-sequence model (Ni and Wang, 2017; Mickus et al., 2019) as does our proposed approach. Ni and Wang propose a sequence-to-sequence model to generate definitions for non-standard English words. Their encoder uses a character-level LSTM to represent the target word and a word-level LSTM to represent the context. An LSTM is also used for decoding. Our proposed approach is similar to that of Ni and Wang, but we do not use an LSTM to encode context, and our LSTM which encodes the target word operates over BPE tokenization as opposed to characters.

An alternative line of research considers definition extraction (e.g., Navigli and Velardi, 2010) in which sentences containing terms and their corresponding definitions are automatically identified in corpora. We focus on definition modelling, as opposed to extraction, because there are very few corpora containing English definitions of Wolasto-
key words on which to apply a definition extraction method, and because Wolastoqey is a polysynthetic language and as such many possible words would not be expected to be found in corpora.

3 Model

Cross-lingual definition modelling can be seen as a machine translation task which involves translating a word in a source language to a definition in some target language. We therefore consider using a network architecture proposed for the task of translation rather than a word-to-sequence model proposed in previous work on monolingual definition modelling (e.g., Noraset et al., 2017). Specifically, we consider a sequence-to-sequence model that makes use of an attention decoder (Bahdanau et al., 2014). We base our model on a sample sequence-to-sequence translation model.\(^2\)

3.1 Encoder Architecture

For our encoder model’s architecture, we use a simple recurrent neural network consisting of an embedding layer followed by a long-short term memory (LSTM) layer. The structure of the encoder is shown in Figure 1.

The embedding layer of our model serves the purpose of representing the meaning of the sub-word tokens that compose our vocabulary. To obtain the embeddings used by our encoder, we consider the approach of initializing the weights of our embedding layer to zeroes as well as the approach of first pretraining our embeddings on a corpus of example sentences extracted from the Passamaquoddy-Maliseet Dictionary. While training, we allow the weights of the embeddings to be updated through gradient descent, but also consider freezing these weights in the case of pretrained embeddings. This is done to further analyze the effects pretraining has on system performance.

At a given timestep, input is passed into our encoder in the form of a sequence of indices corresponding to sub-word representations in our input vocabulary. At a given time-step, the encoder will consider a given input subword. For this subword, the encoder will start by looking up its embedding using the embedding layer. This embedding will then be passed to the LSTM layer. From here, the LSTM layer will use this embedding, the hidden layer of the previous time-step and the context from the previous time-step layer to calculate the value for the current hidden layer which acts as the decision at the current timestep. The context is then updated with the information regarding the current decision, and then passed forward with the output at the current time-step. Once all of the outputs have been calculated, we pass the encoder outputs to the decoder.

3.2 Decoder Architecture

The decoder architecture is shown in Figure 2. Rather than relying on a single vector to contain all information about the input sequence, we instead use an attention decoder to consider the encoder outputs more holistically. Our decoder consists of an embedding layer, two intermediary linear layers, a recurrent LSTM layer and a final linear layer.

\(^2\)https://github.com/spro/practical-pytorch
which we then softmax over to get the output at a given timestep. We apply dropout regularization to the embedding layer in an effort to avoid overfitting.

At a given timestep, input is given to our decoder in the form of the decision made at the previous step, or, in the case of the first decision, a start of sequence token, and will take the form of an index of a token contained within our output vocabulary. We then use this index to look up the embedding for the word through the embedding layer. From here, we concatenate the embedding with the hidden state from the previous decision. We pass this value to the first linear layer which will give us weights we can then compare to the encoder outputs through batch matrix multiplication. This product will then be passed to another linear layer with a ReLU activation function which will set all negative values to 0. This value is then passed as input to our LSTM layer, which will also consider the previous hidden layer and the context from the previous time-step. This will give us updated context and hidden states, which we will pass forward to the next timestep. However, to get the current output, we will need to pass the resulting vector from the recurrent component of our system to another linear layer. The results from this layer will then be softmaxed to give us the final output, and our decision for the current timestep.

3.3 Model Variations

In addition to the base model described above, we also consider two architecture variations designed to analyze the effect of model complexity on performance. Specifically, we consider replacing the unidirectional LSTM layers used in both the encoder and decoder with comparatively simpler GRU layers, and comparatively more complex bidirectional LSTM layers. We consider this because we have a relatively small training corpus which might not be large enough to adequately train more complex models. Although bidirectional LSTMs have been shown to perform better than GRUs and unidirectional LSTMs when sufficient training data is available, they generally perform worse when insufficient training data is available. By considering these model variations, we can attempt to determine whether we have enough training data to justify the use of more complex models.

4 Experimental Setup

In this section, we describe the dataset constructed for these experiments, the evaluation methodology used, and implementation details for our proposed model.

4.1 Dataset

Wolastoqey is a low-resource language. As such, we are limited in regards to our choice of dataset. For cross-lingual Wolastoqey-English definition modelling we require a dataset consisting of Wolastoqey headwords and their corresponding English definitions. The Passamaquoddy Maliseet Dictionary consists of Wolastoqey head words and their corresponding English definitions. Many entries also include parallel Wolastoqey-English example sentences. We use the headwords and definitions to construct our dataset. We use the example sentences to train embeddings, as well as our BPE tokenizer in the case of Wolastoqey.

The Passamaquoddy-Maliseet Dictionary is available online. There is, however, no publicly available download for the contents of this dictionary. We therefore use Selenium, a web automation tool, to crawl the dictionary and extract the entries.

After scraping the dictionary content, we normalize the text. For this, we remove any entries containing errors such as #NAME? as a headword. As each headword can have multiple definitions associated with it, we split definitions on semi-colons as they are used as the primary definition delimiter. We perform a similar operation for the headwords themselves, as an entry can include multiple word-forms for the headword. We split the headword text on commas, with each extracted headword being used to create word-definition pairs with respect to all available definitions for a given word.

Wolastoqey has four parts-of-speech (POS): nouns, pronouns, verbs, and particles (Leavitt, 1996). For this dataset we only include headwords that are nouns, verbs, and particles because there are relatively few entries that are pronouns.

This extraction method produces a dataset that contains 22.5k headword-definition pairs from 19k valid entries. We then split these pairs into training, development, and test sets. To do this, we first group the data based on headwords to prevent any headword with the same form from appearing in more than one of the sets. Once we have grouped the headwords, we then split the data by headword with 80%, 10%, and 10% of headwords in the train-
ing, development, and test sets, respectively. The training set is used for training our model. The development set is used for model tuning. The test set is held out for final evaluation.

In addition to the dictionary headwords, we also extract 18.6$k$ Wolastoqey-English sentence pairs pulled from all valid dictionary entries containing example sentences. As we use these example sentences to train our embeddings, we split this parallel corpus into separate monolingual Wolastoqey and English corpora consisting of 80.5$k$ and 181.9$k$ tokens, respectively.

4.2 Evaluation

Following previous work on definition modelling (Noraset et al., 2017; Ni and Wang, 2017; Gadetsky et al., 2018) we use BLEU score (Papineni et al., 2002) for evaluation. At test time, we generate an English definition for each Wolastoqey headword (in either the development or test set) and calculate the BLEU score between this generated definition and the gold-standard reference definition for this headword.

We compare our proposed model against a baseline that outputs a randomly selected definition from the development set for any input. This baseline will always produce a syntactically well-formed definition, but the definition is unlikely to be semantically appropriate. We consider two variations of this baseline, POS-aware and POS-agnostic, which differ with respect to knowledge of the POS of the input. The POS-agnostic baseline simply outputs a randomly selected definition. The POS-aware baseline outputs a randomly selected definition corresponding to a headword with the same POS as the input. 3

We implement our proposed model using PyTorch 1.7.1. We use a hidden layer size of 256 for each layer in both the encoder and decoder sub-models. Our decoder sub-model’s dropout layer uses a dropout rate of 0.1. To train our model, we use an Adam optimizer with a learning rate of 5e-4. We train all our models for a total of 15000 iterations. We use teacher forcing as our training regimen.

To obtain sub-word representations we use the Huggingface Tokenizers 0.10.1 library. For English, we use the pretrained word-piece DistilBERT (Sanh et al., 2019) tokenizer. For Wolastoqey, we train a BPE tokenizer on the Wolastoqey exam-

ple sentences extracted from the Passamaquoddy-Maliseet Dictionary.

To learn pretrained embeddings we consider both word2vec skip-gram with negative sampling (Mikolov et al., 2013) and fastText (Bojanowski et al., 2017). We use the Gensim version 3.8.3 implementations of skip-gram and fastText. We use the default parameter settings (i.e., a window size of 5 sub-word tokens and 100 dimensional embeddings) except for the minimum frequency to be included in the embedding matrix, which we set to 1 (as opposed to the default of 5) because of the small size corpora we use for training.

We calculate BLEU score using the implementation available in NLTK 3.5 (Bird et al., 2009).

5 Results

In this section we first consider tuning the vocabulary size for the Wolastoqey BPE tokenizer in experiments on development data (Section 5.1). We then present our main results on test data, including an analysis of the impact of pretraining embeddings (Section 5.2) and results for the model variations presented in Section 3.3 (Section 5.3). We then present a qualitative evaluation of the model (Section 5.4).

5.1 Tuning Vocabulary Size

We conduct a grid search to find the optimal vocabulary size for our Wolastoqey BPE tokenizer. The BPE vocabulary will directly affect how our Wolastoqey input words are tokenized and could potentially drastically impact the performance of our proposed model. As we are investigating whether sub-word representations can be leveraged for Wolastoqey-English definition modelling, it is important to determine the optimal vocabulary size.

We consider vocabulary sizes from 2500 to 15000 in increments of 2500. Results on development data are shown in Figure 3. We observe that the optimal vocabulary size is 7500 sub-word tokens. We also observe a steep drop-off in performance when using a vocabulary size that exceeds 7500 tokens. We further observe that the relative performance across parts-of-speech is similar regardless of vocabulary size. We use a vocabulary size of 7500 for the Wolastoqey BPE tokenizer for the remainder of the experiments. At this vocabulary size, each Wolastoqey verb, noun, and particle in the development data is represented by an average of 3.21, 2.56, and 1.93 sub-word tokens,

3 Although we compare against a POS-aware baseline, the proposed model itself has no knowledge of POS.
Figure 3: BLEU score for various Wolastoqey BPE vocabulary sizes on development data.

<table>
<thead>
<tr>
<th>Model</th>
<th>Overall</th>
<th>Verbs</th>
<th>Nouns</th>
<th>Particles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baselines</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POS Agnostic Baseline</td>
<td>0.144</td>
<td>0.156</td>
<td>0.036</td>
<td>0.042</td>
</tr>
<tr>
<td>POS Aware Baseline</td>
<td>0.173</td>
<td>0.184</td>
<td>0.045</td>
<td>0.073</td>
</tr>
<tr>
<td>Base Models</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base Model</td>
<td>0.277</td>
<td>0.304</td>
<td>0.062</td>
<td>0.104</td>
</tr>
<tr>
<td>Base Model (verbs only)</td>
<td>0.306</td>
<td>0.333</td>
<td>0.045</td>
<td>0.050</td>
</tr>
<tr>
<td>Adjustable Pretrained</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>word2vec (Examples)</td>
<td>0.304</td>
<td>0.336</td>
<td>0.080</td>
<td>0.092</td>
</tr>
<tr>
<td>word2vec (News)</td>
<td>0.233</td>
<td>0.257</td>
<td>0.064</td>
<td>0.082</td>
</tr>
<tr>
<td>fastText (Examples)</td>
<td>0.299</td>
<td>0.327</td>
<td>0.058</td>
<td>0.085</td>
</tr>
<tr>
<td>Frozen Pretrained</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>word2vec (Examples)</td>
<td>0.148</td>
<td>0.156</td>
<td>0.041</td>
<td>0.070</td>
</tr>
<tr>
<td>fastText (Examples)</td>
<td>0.209</td>
<td>0.223</td>
<td>0.045</td>
<td>0.045</td>
</tr>
</tbody>
</table>

Table 1: BLEU score for each model overall and for each POS.

respectively.

5.2 Effects of Pretraining

Results on test data for the proposed model and the baselines are shown in Table 1. We first consider the baselines. The POS-aware baseline, as expected, outperforms the POS-agnostic baseline, overall and for each POS.

The base model (i.e., the model proposed in Sections 3.1 and 3.2) outperforms both baselines. This finding demonstrates that sub-word representations of Wolastoqey words based on BPE can be leveraged to generate English definitions. We note that the performance is much better on verbs, which are the most common POS in our datasets, than other parts-of-speech. We hypothesize that this is because Wolastoqey verbs often consist of multiple morphemes, and indeed are on average split into more sub-word units than other parts-of-speech in the analysis in Section 5.1, whereas other parts-of-speech often correspond to a single morpheme, and are split into fewer sub-word units. We consider training the base model on only verbs, shown as “Base Model (Verbs Only)” in Table 1. Here we see a slight improvement in performance over the base model on verbs, and a corresponding reduction in performance on other parts-of-speech.

The base model does not use pretrained embeddings. We now consider experiments using pretrained embeddings, in which we allow the embeddings to be adjusted through updating during training (“Adjustable Pretrained” in Table 1). We consider word2vec and fastText embeddings trained on the Wolastoqey and English corpora built from the example sentences in the Passamaquoddy-Maliseet Dictionary (“word2vec (Examples)” and “fastText (Examples)”, respectively). Both of these approaches improve over the base model in terms of overall BLEU score. For English, because it is a high-resource language, we have access to many sources of embeddings which are pretrained on much larger corpora. We therefore consider using English word2vec embeddings pretrained on text from Google News, shown as “word2vec (News)”. This requires switching from word-piece tokenization to word-level tokenization for English. For Wolastoqey, we still use BPE tokenization and train on the corpus of Wolastoqey example sentences. Although these English embeddings are trained on a much larger corpus, this does not yield an improvement over using embeddings pretrained on the English example sentences.

Finally, we consider the impact of allowing the embeddings to be updated during training. We again consider word2vec and fastText trained on the corpora of Wolastoqey and English example sentences, but here we freeze the embedding weights when training the model (“Frozen Pretrained” in Table 1). These methods perform poorly compared to the base model, and compared to the case where the embeddings are updated during training. In particular, here the word2vec embeddings perform roughly on par with the POS agnostic baseline. These findings indicate the importance of allowing the embeddings to be updated during training.

The base model substantially outperforms both baselines considered. In the following subsection we consider further variations on the base model.

5.3 Model Variations

Table 2 shows results on test data for the base model using a (unidirectional) LSTM (i.e., the base model
presented in Table 1) and GRU, and a bidirectional LSTM. We observe that using a GRU in-place of an LSTM gives a better overall BLEU score. Despite being more powerful, the bidirectional LSTM performs worse overall than the unidirectional LSTM base model. We hypothesize that, because of the relatively small size of the training data, simpler models, such as a GRU, can be more effectively trained. This finding, combined with the findings from Section 5.2, suggests that there could be scope for further improvement through the use of pre-trained embeddings with a GRU.

5.4 Qualitative Analysis

While BLEU score provides a method of empirically evaluating our system, we also wish to perform a qualitative analysis of our system’s outputs. For this analysis, we generated definitions for 20 randomly-selected test set Wolastoqey words and manually compared the generated definitions to their ground-truth reference definitions. This analysis was carried out by the first author of this paper, an English first language speaker and Wolastoqey learner. We analyzed the definitions with respect to both semantics and syntax.

For semantics, we considered whether the generated definitions were topically-related to the reference definitions. Of the 20 definitions, 10 were determined to have little to no topical relatedness to the reference definition, 8 showed some level of topical relatedness to their ground truth reference, and 2 were determined to be reasonable definitions for their respective words. For this analysis, we consider reasonable definitions to be definitions that contain few or no syntactic errors and do not significantly vary in meaning when compared to their ground truth references. An example of a word our system is able to reasonably define with little error is ‘t-uwapolokehkimal, which the Passamaquoddy-Maliseet Dictionary defines as ‘s/he instructs h/ incorrectly; s/he teaches h/ incorrect information, etc.’ For this word, our system generates the definition ‘s/he teaches h/ incorrectly’. An example of a generated definition that shows some level of topical relatedness to the reference definition can be seen for the verb kcitawse, for which our system generates the definition ‘s/he walks without walking, walks in’ whereas the reference definition is ‘s/he walks far into or sinks into, s/he gradually works way into’. In this example, both definitions share some reference to the action of walking; however, the meaning of the generated definition deviates from its ground truth reference.

Considering syntax, we observed that 13 out of the 20 definitions generated demonstrated correct syntactic form and were overall comprehensible output sequences.

6 Conclusions

In this paper, we considered cross-lingual Wolastoqey-English definition modelling, in which we automatically generate English definitions for Wolastoqey words. Our work is in contrast to most prior work on definition modelling which has been monolingual, i.e., the word being defined and its definition are in the same language. In further contrast to most prior work on definition modelling, where Wolastoqey is a low-resource language, we do not have access to a large Wolastoqey background corpus for training. We showed that a sequence-to-sequence model that represents morphologically-complex Wolastoqey words at the sub-word level using BPE segmentation outperforms baseline approaches. We further demonstrated that the proposed approach can be improved by pretraining on small Wolastoqey and English monolingual corpora built from dictionary example sentences and by using a GRU instead of LSTM. Qualitative analysis revealed that the generated definitions are often syntactically well-formed and topically related to the gold-standard reference definitions.

In future work we plan to investigate alternative strategies for representing Wolastoqey words in the encoder, including character-level approaches and segmentations based on unsupervised approaches to learning morphology (Creutz and Lagus, 2002). We also plan to explore alternative model architectures, including transformer-based models (Vaswani et al., 2017) and models that incorporate large pre-trained English language models (e.g., Lewis et al., 2020).

Ethical Considerations

Wolastoqey is an Indigenous language and natural language processing can reinforce colonialist views
(Bird, 2020). The first author of this paper is Wolastoqew. The Passamaquoddy-Maliseet dictionary can be used for research purposes. We obtained permission to scrape the dictionary content for use in natural language processing research.

References


Neural Network-Based Generation of Sport Summaries: a Preliminary Study

David Stéphane Belemkoabga*, Aurélien Bossard**, Abdallah Essa*, Christophe Rodrigues* and Kevin Sylla*

*Léonard de Vinci Pôle Universitaire, Research Center
92 916 Paris, La Défense, France,
firstname.name@devinci.fr

**Laboratoire d’Informatique Avancée de Saint-Denis, Université Paris 8
2 rue de la Liberté, 93526 Saint-Denis, France
firstname.name@iut.univ-paris8.fr

Abstract
This paper presents a global summarization method for live sport commentaries for which we have a human-written summary available. This method is based on a neural generative summarizer. The amount of data available for training is limited compared to corpora commonly used by neural summarizers. We propose to help the summarizer to learn from a limited amount of data by limiting the entropy of the input texts. This step is performed by a classification into categories derived by a detailed analysis of the human-written summaries. We show that the filtering helps the summarization system to overcome the lack of resources. However, several improving points have emerged from this preliminary study, that we discuss and plan to implement in future work.

1 Introduction
In this paper, we propose a new approach for sport commentary summarization. This approach is still an ongoing work, and the results presented here are preliminary. Sport commentaries represent an interesting resource, as the live commentaries we work on are associated with a summary written by an expert: the commentator himself. Indeed, the commentator writes a summary at the end of every game. Automatically generating a game summary would release the commentator from a part of his heavy workload and thus would free up his time for more complex and rewarding tasks, such as game in-depth analysis.

Summarizing live sport events commentaries is a challenging task. First of all, they are live written; new commentaries can conflict with or complete former ones. For example, if a soccer player scores two goals, the live commentaries about each goal are relevant information. However, extracting the live commentaries about each goal will not be sufficient in order to generate a good summary. It would indeed lead to producing redundant summaries.

Moreover, if you consider a game as an event, it is composed of several subevents. Some of them are deemed important enough to be commented. However, most of those commented subevents are not important enough to appear in a summary. So, live commentaries are mostly made up of noise: in a soccer game, there will be more shots than goals, even more substitutions than goals, which are the most important information of a game. This noise has to be filtered. Therefore, studies have to be carried out about the relevance of an information in the game summarization context.

The style of human-written summaries differs from the one of live commentaries. For all these reasons, statistical extractive summarization models are not relevant for this kind of data. Extractive models consist in fact in extracting relevant sentences from source corpora and put them together in order to build a summary. The difference in style between summaries and commentaries and the noise in commentaries are a substantial hindrance to building live commentaries summaries with extractive models. As for neuronal abstractive models, fast-growing these past few years, they need huge training corpora to be efficient: several hundreds of thousands of documents associated to their summary. However, we can only assemble a corpus that covers five years of a national championship – approximately 1700 game commentaries associated with their summary.

Moreover, neural abstractive summarizers are mostly designed for news summarization. News summarization fits well neural summarizers as neural models can only take a limited number of words for input. Journalists use an inverted pyramid structure, so the most important information is packed in the first paragraph. Taking the first n words as input ensures that a neural model will work only with
important information. Live commentaries do not have such a hierarchical structure, and commonly used statistical indicators do not seem to be useful. Therefore, selecting the input of a neural model is a challenge in our application.

In order to overcome these obstacles, we need to train our own abstractive summarization model. We have to reduce the data noise in order to allow the abstractive model to converge quickly with a limited amount of data. We hypothesize that the human summaries variability is low enough to make it possible for the model to learn despite having a small corpus as input.

We propose a global summarization method that aims to lower the input data entropy in order to enhance the automatic summaries quality. This method relies on an information selection prior to learning in order to shorten the input texts and thus get rid of data that are useless to the summary generation.

The paper is structured as follows: in a first part, we present the related work. In a second part, we introduce our corpus and its features. Then we describe our method, followed by the experiments and the results. We end by discussing the results and exposing our perspectives.

2 Related Work

The automatic generation of sports commentary summaries is, to our knowledge, a subject that is very little discussed in the literature.

We can mainly cite the work of (Zhang et al., 2016) which, from game commentaries available online, generates an extractive summary. The method consists of three main steps: a first step of modeling sentences according to surface clues defined empirically as the sum of the \( tf-idf \) sentence words, the presence or absence of important words in the sentence such as “red card”, ”goal”... Then, from a set of reference sentences, a learning step predicts the sentences ROUGE score according to these surface clues. Finally, a last ranking step allows the sentences with the best scores to be incorporated into the summary. According to these authors, however, the approach suffers from several limitations. Since the process is sentence-centered, the summaries generated have a lot of redundancy between sentences. Furthermore, learning tends to penalize short sentences that are sometimes wrongly considered less informative because their direct contribution to the ROUGE score is lower.

On this same task, we can also cite the work of (Bouayad-Agha et al., 2012) whose particularity is to propose a system of generative summary based on the definition of a specialized ontology for soccer games. Thus, from the data extracted from commentaries in an ontology, handwritten rules are triggered in order to rephrase the information and generate a summary. The main limitations of the approach are the need for an exhaustive ontology population (players, teams,...) as well as a generation of stereotypical summaries because they are built from the same rules.

On the same issue but from a very different angle, the work of (Corney et al., 2014) starts from the comments of twitter users during games and produces subjective summaries. For each official commentary related to an event during the game, the supporters’ comments on twitter are analyzed on a 4 minutes window, the goal being to extract the most representative tweet of this event from the subjective point of view of the supporters of each team. For that purpose, the tweets are first distributed between the two teams. A user is defined as a supporter of a team if in his comments the team is overrepresented compared to other teams. In a second step, for each team, the most important topics are detected using a variant of the \( tf-idf \). Finally, for each team, the most representative tweet of the subjects found is selected, without further processing.

More recently, the work of (Li et al., 2019) presents a model able to produce NBA games summaries. Based only on game statistics, it can generate a summary composed of two parts: a game overall summary and a player centered summary. This model uses latest deep learning techniques with a Wasserstein generative adversarial networks (WGAN) proposed by (Arjovsky et al., 2017). However, despite the model used, the method only generates stereotyped summaries filling a fixed template. For the overall summary, the template is defined as follows:

\[
\text{On [Date], [Team(A)] made a [Score(A)]-Score[B] [learned phrase] [Team(B)].}
\]

Except for the learned phrase, all elements in [ ] are directly assigned from game statistics. The learned model will only affect the learned phrase used to characterize with words to which degree team A wins or loses against team B.
To avoid the pitfall of generating stereotypical summaries, given the difference in style between the summaries and the live commentaries, the high noise level in the commentaries and the low volume of data, we choose to stand out from previous methods by taking advantage of recent advances in neuronal generation. To do so, we approach the problem from the perspective of neuronal generation preceded by a noise reduction step in the source texts to allow the generative model to converge quickly.

3 Live Commentaries Corpus

We extracted all the available Ligue 1 soccer games commentaries and their associated human-written summary from L’Equipe website (http://www.lequipe.fr). The archived games with live commentaries cover a five seasons period from 2014 to 2019. Game commentaries are 8000 words long on average and 80 interventions of the commentator. Live commentaries about the game are interspersed with facts about the game players: information about a recent trade, an ongoing goals streak... Manual summaries are 55 words long on average (cf. Figure 1). Therefore, live commentaries are significantly longer than 400 words which is the length commonly used by most neuronal summarization models in order to reduce their complexity (Rush et al., 2015; See et al., 2017).

The complexity of automatic summarization based on such live commentaries is thus far too important, especially given the small amount of documents we can use to learn.

Figure 1 shows the three last minutes of Paris vs Lille commentaries. It displays the noise in these documents. One can especially notice a poll posted by the commentator between 90+1’ and 90+2’ minutes, and three commentaries that one can consider as noise for the purpose of generating a short summary: extra time announcement and three missed actions. One can also notice the difference in style between the summary and the commentaries.

The specificities of these documents force us to rethink the automatic summarization process by first filtering only relevant information, then generating a summary in order to mimic manual summaries style.

4 Our Model

Automatically summarizing soccer game commentaries presents a major difficulty: standard frequency-based techniques for evaluating the importance of a word or sentence are ineffective due to the source documents specificity: the most important information about a game is often the rarest one – result, expulsions, goals. This, combined with the style of the summary which is radically different from the commentaries, led us to abandon the extractive methods for an abstract method. However, the small number of documents that can be used for learning is very restrictive1.

1Even if we had summaries over twenty years, we would have 500 times fewer documents than the CNN/Dailymail Corpora used in most abstractive neural summarization research works.
We hypothesize that, given the small average size of summaries and their low linguistic variability, the decoder part of an encoder/decoder model can learn how to generate a summary using style elements of manual summaries. On the other hand, the relatively large size of the commentaries makes the encoding task more complicated, if not impossible given the small size of the learning corpus.

Therefore, we choose to reduce the entropy of the source texts in order to allow an encoder/decoder to learn how to generate summaries with a restricted corpus size. The simplest idea is to keep in the input commentaries only those that are deemed relevant for the development of a summary, before learning an automatic summary model based on these filtered commentaries. We detail both steps here.

4.1 Sentence filtering

In order to filter the input sentences, we need to characterize the sentences that carry important information, and those that do not. To this end, we decided to rely on the manual summaries written by the game commentators. We assume that these summaries contain only relevant information.

4.1.1 Manual corpus annotation

We analyzed an entire year of League 1 (so 380 pairs of live commentaries / summary) and typed the information found in the summaries. The presence of a particular information within a summary is a sign of its relevance. We therefore identified the types of information, then counted the occurrences of the different types of information and kept the most frequent ones.

This led us to the following list of information categories, summarized in Table 1.

Table 1: Information categories and the percentage of summaries in which they are represented (based on an entire League 1 season)

<table>
<thead>
<tr>
<th>Information type</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>80</td>
</tr>
<tr>
<td>Championship rank</td>
<td>55</td>
</tr>
<tr>
<td>Goal scorer</td>
<td>45</td>
</tr>
<tr>
<td>Team domination</td>
<td>24</td>
</tr>
<tr>
<td>Win/loss streak</td>
<td>22</td>
</tr>
<tr>
<td>Efficiency</td>
<td>19</td>
</tr>
<tr>
<td>1st/2nd half quality</td>
<td>18</td>
</tr>
<tr>
<td>Game quality</td>
<td>18</td>
</tr>
<tr>
<td>Ejection</td>
<td>16</td>
</tr>
<tr>
<td>End of w/l streak</td>
<td>14</td>
</tr>
<tr>
<td>Missed penalty</td>
<td>7</td>
</tr>
<tr>
<td>Balanced game</td>
<td>5</td>
</tr>
<tr>
<td>Converted penalty</td>
<td>4</td>
</tr>
<tr>
<td>Injury</td>
<td>3</td>
</tr>
<tr>
<td>1st game since</td>
<td>3</td>
</tr>
<tr>
<td>Player missing</td>
<td>3</td>
</tr>
<tr>
<td>Decisive coaching</td>
<td>3</td>
</tr>
<tr>
<td>Missed penalty</td>
<td>7</td>
</tr>
<tr>
<td>Balanced game</td>
<td>5</td>
</tr>
<tr>
<td>Converted penalty</td>
<td>4</td>
</tr>
<tr>
<td>Injury</td>
<td>3</td>
</tr>
<tr>
<td>1st game since</td>
<td>3</td>
</tr>
<tr>
<td>Player missing</td>
<td>3</td>
</tr>
<tr>
<td>Decisive coaching</td>
<td>3</td>
</tr>
</tbody>
</table>

Then, we systematically searched for this information in the live commentaries and annotated them according to the type of information they conveyed. For example, the figure 1 does not contain a relevant commentary, but the game summary contains important information: Neymar’s return, PSG’s efficiency (“surgical, the Argentinians...”), Icardi’s goal, Di Maria’s goal, PSG’s victory (resulting from the half-time win and subsequent management). In this summary as in many others, information is implied and derived from other information, which has made the task of defining types of information particularly complex. Thus, we have an annotated corpus to learn to categorize commentaries according to the information they carry.

4.1.2 Categorizing Commentaries

We kept only the 17 most frequent classes, considering that below a certain threshold – empirically set – the frequency of a type of information within the summaries was too low for it to be considered important.

Before proceeding with the sentences classification, we trained a language model (Bengio et al., 2003) (Sundermeyer et al., 2015) on the commentaries corpus in order to take into account the specificities of this particular corpus (specific vocabulary, different style from the general language). This model, represented in figure 2 learns word embeddings thanks to a neural network of bi-LSTM (Graves and Schmidhuber, 2005) units which aims at improving the next-word probability prediction.

Figure 2: Language model architecture

We then proceeded to train a binary classification model of commentaries on a one-year sample of annotated commentaries. The model used represented in figure 3 is a Bi-LSTM (two successive layers of LSTM, one proceeding from the beginning to the end of the sentence, the other from the end to the beginning of the sentence). This bi-directional architecture allows better results in
language processing tasks. The input layer takes a game commentary and the output layer a binary value. The commentary is classified as relevant if it covers one of the 17 selected categories, otherwise the game is classified as irrelevant.

![Figure 3: Classification model architecture](image)

We applied this model to all commentaries outside the learning corpus. As a result, we can filter and present to a neural summary model only the commentaries deemed relevant, and thus improve the model response by reducing the entropy of the input data. The results of the binary classification are presented in Table 2.

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary model</td>
<td>0.87</td>
<td>0.89</td>
<td>0.87</td>
</tr>
</tbody>
</table>

Table 2: Accuracy of the classification model

4.2 Generative summary model

We used a pointer-generator network (See et al., 2017). We trained it on two different datasets: a Raw corpus and a filtered corpus (binary classes). Pointer-generator is a supervised learning method derived from sequence-to-sequence translation models (Bahdanau et al., 2014) with an attention mechanism (Nallapati et al., 2016).

5 Experiments

To test our approach, we compare two automatic summary models: an extractive method, TextRank (Mihalcea and Tarau, 2004) as well as a generative method, pointer-generator on the sample of the last 167 League 1 games that were not used during any learning step.

**Experimental Setup:** Both methods are tested with and without prior filtering of the sentences judged relevant, according to the method presented in §4.1.2, in order to validate the hypothesis that reducing entropy in source texts has a positive effect on model convergence and on the quality of the summaries produced. The workflows for generative methods with and without pre-filtering are shown in Figures 4 and 5.

![Figure 4: Pipeline architecture without classification](image)

The language model uses a word embedding layer of dimension 64. The recurrent cells of the encoder and decoder are of dimension 64. The size of the output layer is the size of the vocabulary, which is 4480.

The classification model uses the same word embedding layer that it retrieves from the language model after training and two layers of LSTM (bi-LSTM) each of size 16, the output layer is of size 2 (0 or 1 for important and unimportant).

The pointer-generator model uses an encoder and a decoder with bi-LSTMs of dimension 128. The vocabulary size of the model is 50000. During training, the model takes texts truncated to 400 words and produces summaries of no more than 100 words, which is much more than the number...
of words used in human summaries. In order to reduce the size of the problem and speed up training, we have a batch size of 4. Error backpropagation is done with the Adam optimizer with a learning rate of 0.15. The models were learned over 30000 iterations (80 epochs). It takes fewer iterations than the See et al. (2017) model to get results because our training set is much smaller.

**Evaluation Metric:** Summaries are evaluated with the commonly used ROUGE package (Lin, 2004). The ROUGE-N score is a metric that compares the N-grams in common between the reference summary and the summaries to be evaluated. We took as a reference summary the summaries written by the commentator at the end of the game.

We use the specific configuration that showed the best correlation with human evaluations in Graham (2015) (a ROUGE-2 precision score).

**Baselines:** We compare the generative method to two extractive methods: one without, and one with filtering commentaries. We use TextRank (Mihalcea and Tarau, 2004), a method comparable to (Radev, 2004) but designed for mono-document summarization. It is a graph-based method that considers summarization as the extraction of the most central sentences in a graph. The implementation used is the one of Nyzam and Bossard (2019), freely available online\(^2\). Even if TextRank method was designed in 2004, it was shown in (Zheng and Lapata, 2019) that it still compares to more recent methods when there is no correlation between sentence position and centrality.

### 6 Results

The results are presented in the table 3. We observe a consequent improvement in the ROUGE scores of the extractive and generative models when run on filtered commentaries.

We also find that extractive models are better in recall but less accurate than generative models. The extractive model used here indeed maximizes the number of words in the abstracts, unlike the generative model. As a result, extractive summaries are composed of 69 words on average compared to 44 words for generative summaries, so they can carry more information.

It is also noted that while the generative summaries present relevant information, it is often poorly expressed. Figure 6 shows an example of a summary generated by our system (we try to retranscript syntactic errors in our translation in italics)

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extractive</td>
<td>3.5</td>
<td>1.7</td>
<td>2.3</td>
</tr>
<tr>
<td>Extractive + filtering</td>
<td>3.7</td>
<td>2.4</td>
<td>2.7</td>
</tr>
<tr>
<td>Generative</td>
<td>2.5</td>
<td>3.4</td>
<td>2.9</td>
</tr>
<tr>
<td>Generative + filtering</td>
<td>2.9</td>
<td>4.1</td>
<td>3.3</td>
</tr>
</tbody>
</table>

**Table 3:** ROUGE-2 Scores of the different summary systems. The best score obtained with Graham (2015)’s configuration is in bold.

Extractive summaries contain a great deal of redundancy and irrelevant information. Using surface clues other than purely frequency clues, such as (Zhang et al., 2016), would surely partly solve this problem. Figure 8 presents an extractive summary for the same game as the summaries of Figures 6 and 7.

### 7 Discussion

Our model offers relevant information, but with an approximate linguistic quality. We assume that

---

\(^2\)MOTS : [https://github.com/ToolAutomaticSum/MOTS/](https://github.com/ToolAutomaticSum/MOTS/)

---

**Figure 6:** Example of a summary generated by our generative system (we try to retranscript syntactic errors in our translation in italics)

**Figure 7:** Example of a manual summary written by L’Equipe’s commentator (our manual translation in italics)
Figure 8: Example of an extractive summary (manual translation in italics)

Sala scores a brace! Thomasson keeps pushing on the left of the box but gets blocked by a LOSC player. Lucas Lima gets the ball and crosses with his left foot. Free at the 6-yard line, Sala heads the ball and beats Maignan on his left side. Sala forward. © L’Equipe Thanks to his scorer Sala (9th goal in L1 this season), the FC Nantes leads unsurprisingly after the first half as they were solid and efficient. Sala scores! He crosses for Thomasson who passes towards Sala. Sala forward

this lack of linguistic consistency is mainly due to the lack of training data; it prevents the model from capturing language features despite the low linguistic variability displayed in human-written summaries due to document specialization.

As we assumed, the extractive model results in very noisy juxtapositions of commentaries that contain a lot of irrelevant information. The results are therefore far away from human summaries. This is shown on the one hand by reading the summaries produced and on the other hand by the evaluation of the ROUGE score.

Commentary filtering prior to learning does improve the quality of the generated summaries. Learning is simplified by reducing noise and the size of the input data.

We did not perform a manual evaluation of the summaries, because the work is still in its early stages and the time needed for a manual evaluation, e.g. pyramid (Nenkova and Passonneau, 2004) is better spent in late stages. However, we analyzed the automatic summaries produced by our method, and we made the following observations:

- they are close to be grammatically correct;
- even if their ROUGE-2 scores are twice as good as TextRank model, they lack some major information;
- linguistic quality of the results with pre-filtering is far better than without pre-filtering (observation that needs to be confirmed by an accurate evaluation).

This can be due to several causes: first, even if we filtered the commentaries, they are still longer on average than the 400 words commonly used as input of neural summarizers (and used by our summarizer as well). Given the small amount of data available for training and the fact that information is cut from the input texts, it can explain that major information is missing.

Second, the filtered commentaries are still noisy. Instead of using filtering techniques, information extraction techniques could be used to fill predefined templates for the 17 important information categories we defined in §4.1.1. This would lead to more concise input texts, focusing on the core of each relevant information only.

Third, some information cannot be found in the commentaries. We think of championship ranks, ongoing streaks, which are rarely raised in pre-game commentaries, or the overall technical quality of a game, which can be derived from game statistics (percentage of completed passes). However, these statistics can be extracted and given as input of the pointer-generator decoder. This way, the pointer-generator would have access to the information needed to generate sentences conveying what our analysis of the data considered as major information.

8 Conclusion

In this paper we presented a model that allows the generation of abstractive summaries of specialized documents with limited data in French language. Our goal was to show that for summary generation and in specific contexts, abstract models could converge more quickly by reducing the entropy of the input data. Our preliminary results show that after having filtered the input texts and even with a small amount of data, the neural summarizer reaches a much higher precision, and also a better linguistic quality.

We found that much of the information needed to manually generate summaries is not present in the live commentaries. Indeed, many important facts: absence of a player, efficiency, domination of a team, balanced game, are often only deductible from non textual data. Systematically providing this input data to a generative system can help it to improve summary generation. In this way, we plan to add to the text sequences the relevant statistics for the generation of summaries. We also plan to provide more focused and concise texts as input to a neural generative summarizer in order to improve its summaries even with a limited amount of data. We could also improve a language model by using extra texts about soccer games, and thus improve the linguistic quality of the generated summaries.
Acknowledgement

This work is supported by a public grant overseen by the French National Research Agency (ANR) as part of the “Young researchers program” (reference: ANR-16-CE38-0008 – ASADERA).

References


Abstract

Split-and-rephrase is a challenging task that promotes the transformation of a given complex input sentence into multiple shorter sentences retaining equivalent meaning. This rewriting approach conceptualizes that shorter sentences benefit human readers and improve NLP downstream tasks attending as a preprocessing step. This work presents a complete pipeline capable of performing the split-and-rephrase method in a cross-lingual manner. We trained sequence-to-sequence neural models as from English corpora and applied them to predict the transformations in English and Brazilian Portuguese sentences jointly with BERT’s masked language modeling. Contrary to traditional approaches that seek training models with extensive vocabularies, we present a non-trivial way to construct symbolic ones generalized solely by grammatical classes (POS tags) and their respective recurrences, reducing the amount of necessary training data. This pipeline contribution showed competitive results encouraging the expansion of the method to languages other than English.

1 Introduction

Text Simplification (TS) is the process of modifying natural language to reduce complexity and improve both readability and understandability (Shardlow, 2014). A simplified vocabulary or a simplified text structure can benefit people with limited language skills, such as those with low education levels, children, non-native speakers, and individuals with learning impairments (e.g., autism, dyslexia, or aphasia) (Štajner et al., 2015; Guo et al., 2018). Furthermore, when applied as a preprocessing step, TS may also improve the performance of several natural language processing (NLP) tasks, such as parsing, machine translation, semantic role labeling, text summarization, information extraction, among others (Niklaus et al., 2019a; Štajner and Popović, 2019).

Most work on TS has concentrated on analyzing specific characteristics at the sentence level, fashioning the task of sentence simplification (SS) (Alva-Manchego et al., 2020). SS applications aim to identify and solve two main aspects: lexical complexity, which refers to difficult words or expressions in the text (e.g., non-frequent words, specific terminologies, foreign words, etc.) (Štajner et al., 2020; Narayan and Gardent, 2014); and syntactic complexity, which refers to the length of the sentences and their grammatical complexities (e.g., number of subordinate or coordinate clauses, unusual sentence structures, depth of the syntactic tree, among others) (Štajner et al., 2020; Rebello et al., 2019).

Split-and-rephrase, proposed by Narayan and co-authors (Narayan et al., 2017), is a novel sentence simplification task that has attracted much research interest in the NLP field. Its goal is to split and rephrase a complex input sentence into shorter sentences that retain equivalent meaning (see examples in Figure 1). Neither deletion nor lexical/phrasal simplification is intended. The core of this process is to properly make the syntactic transformations required by the split action (e.g., turn a relative clause into a main clause).

Figure 1: Basic split-and-rephrase examples highlighting the transformations promoted by the split action.

This bottle was used until 2002 when it was dropped in favor of a traditional bottle.

He sought medical care in Rome, but it was unsuccessful, and he died at the age of 42.
This work innovates from previous split-and-rephrase methods. We present a complete pipeline capable of performing the split-and-rephrase challenge by combining trained sequence-to-sequence neural models that rely on symbolic vocabularies accompanied by BERT’s masked language modeling. The main contribution is to construct a cross-lingual solution that deals both with English and Portuguese sentences. In addition, we enhanced a preliminary work (Berlanga et al., 2020) promoting analysis against complete reference test sets and comparing results to similar models/pipelines. To the best of our knowledge, this is the first complete pipeline to address split-and-rephrase in a cross-lingual manner, encouraging the expansion of the method to languages other than English.

2 Related Work

As discussed by Narayan and colleagues (Narayan et al., 2017), split-and-rephrase method must be distinguished from other sentence rewriting tasks, such as sentence compression, sentence fusion, and sentence paraphrasing. Furthermore, in contrast to the conventional sentence simplification task, split-and-rephrase does not entail loss of information, thus targeting the meaning preservation despite the split behavior (Alva-Manchego et al., 2020).

In an observational study, Gasperin and colleagues (Gasperin et al., 2009) stated that sentence splitting was the most frequent syntactic simplification operation used by an annotator when creating simplified texts. Among the techniques to perform the transformations required by sentence splitting, Niklaus et al. (Niklaus et al., 2019a) segregates them into three classes: (a) Syntax-driven rule-based approaches that use a set of hand-written rules to detect points where sentences may be split (Siddharthan and Mandya, 2014; Ferrés et al., 2016); (b) Semantic parsing based approaches that aim to decompose sentences into minimal semantic units that may be split into individual output sentences (Narayan and Gardent, 2014; Sulem et al., 2018); and (c) Data-driven approaches where the splitting point and transformations are learned automatically from training in aligned corpora of complex-simple sentences (Narayan et al., 2017; Aharoni and Goldberg, 2018).

Concerning split-and-rephrase previous works, Narayan et al. (Narayan et al., 2017) recently presented data-driven baseline models to help with some insights about the task, together with the WebSplit benchmark corpus. After that, Aharoni and Goldberg (Aharoni and Goldberg, 2018) established more robust baselines augmenting sequence-to-sequence neural models with copy-mechanism (Gu et al., 2016), and also released an updated version of WebSplit to reduce overlap in the data splits. Given the small vocabulary and the unnatural linguistic expressions present in WebSplit corpora. Botha et al. (Botha et al., 2018) compiled the WikiSplit corpus uniting more than one million naturally occurring sentence rewrites obtained from mining English Wikipedia’s edit history. Later, Niklaus et al. (Niklaus et al., 2019b) constructed the MinWikiSplit corpus running DisSim framework (Niklaus et al., 2019a) over the WikiSplit data and applied a set of 35 handwritten transformation rules to decompose source sentences in more split simplified counterparts.

As for the Portuguese language’s split-and-rephrase task, based on the literature surveyed, we found no specific corpus built for this purpose. However, Leal et al. (Leal et al., 2018) made available the PorSimplesSent data set, a Brazilian Portuguese corpus to study sentence readability assessment, which we incorporated into this work to further test our pipeline.

3 Methodology and Data

We define the split-and-rephrase task as follows. Given a complex sentence \( C \), the goal is to produce a simplified text \( T \) consisting of a sequence of sentences \( T_1, T_2, \ldots, T_n, n \geq 2 \), in such a way that \( T \) preserves the meaning of \( C \).

In this section we specify the details about the implementation of our proposed pipeline and all the above mentioned split-and-rephrase corpora employed in this work.

3.1 Pipeline Specification

Our complete pipeline is composed of two main elements: (1) one trained sequence-to-sequence neural model that relies on a given custom symbolic vocabulary explained ahead; and (2) the BERT’s masked language modeling. The overview of the pipeline is illustrated in Figure 2. Below we present these elements and how they are integrated.

**Sequence-to-sequence neural models** Our constructed models were based on the conventional encoder-decoder architecture composed of Gated Recurrent Unit (GRU) neural networks with attention mechanism (Bahdanau et al., 2014; Cho et al.,
Figure 2: Illustration of our complete pipeline. To perform a prediction in the pipeline, the complex input sentence (A) passes through a preprocessing step to convert the text into symbolic vocabulary (B). This converted symbolic sequence is given to a sequence-to-sequence neural model that produces an output based on the learned knowledge on how to split such items (C). The model’s output symbolic sequence is then reconverted to genuine text (D) and fed into BERT’s masked language modeling to generate the simplified output sentences filling eventual gaps (E).

2014). Such mechanism makes it possible to establish references at particular points in original sequences, and enable the transmission of these instances to the decoder outputs. This approach is known to be an appropriate strategy for training models in aligned corpora and has shown excellent results for text-to-text NLP tasks (Raffel et al., 2019). The attention layer is connected to the encoder-decoder GRU layers, both composed of 100 units. We employed a batch size of 200 and the training process lasts 10 epochs in Training Setup 1 and 40 epochs in Training Setup 2. These two distinct setups are further discussed in Section 4. We used categorical-cross entropy loss function and applied Adam optimization algorithm (Kingma and Ba, 2014) to update the networks’ weight iteratively.

Symbolic vocabulary Contrary to traditional approaches that seek training models with extensive genuine vocabularies, we feed our sequence-to-sequence neural models with custom symbolic ones generalized uniquely by the concatenation of grammatical classes (POS tags) and their respective recurrences (indexes) observed in the aligned sentence pairs from the training data sets. The wildcard character ‘*’ is used for padding. The custom implementation to build such vocabulary is illustrated in detail in Figure 3. We found this strategy drastically reduced the vocabulary size to only a few items optimizing training process times.

This symbolic vocabulary approach is the key factor that enables our models to work in a cross-lingual manner: instead of dealing with genuine texts, they are capable to understand items in common gained from sentences of different languages, namely English and Portuguese, given that features such as grammatical classes are standard across these languages (Stodden and Kallmeyer, 2020). In addition, due to the existence of syntax-based patterns behind splitting in both languages, the specific knowledge on how to split the sentences may be captured accordingly thanks to the sequential observation nature of sequence-to-sequence neural models with attention mechanism.

BERT’s masked language modeling Since our models are trained with alignments of symbolic sequences (such the example in Figure 3), they may predict symbolic sequences of items that need to be reconverted to genuine texts. But such predicted items are not always present in the complex input sentence to be converted back (see example in Figure 4). To fill this gap, we employed BERT’s masked language modeling (MLM) (Devlin et al., 2018). BERT is proposed to train deep bidirectional language representations based on the Transformer architecture (Vaswani et al., 2017). Instead of predicting the next word in a sequence given the history, MLM predicts missing tokens in a sequence given its left and right context (Qiang et al., 2020). For the English language, we adopted the pre-trained model on English Wikipedia and Book Corpus. For the Brazilian Portuguese language, we employed the large trained model from BERTimbau work (Souza et al., 2020).
Anaya was born in Palencia, Spain, and is the youngest of three children.

The building was then turned into a railway heritage centre in 1979 by the Butetown Historic Railway Society.

In 1994 the railway started to run steam hauled passenger services up 500 m of track.

WebSplit v0.1 Narayan et al. (Narayan et al., 2017) launched this corpus as the first data set to address the split-and-rephrase task. It is composed of 1,100,166 sentences written from RDF tuples. Due to the fact that one single complex sentence may map to a set of \( S_n \) structurally simplified references, the actual number of distinct complex sentences, \( |C| \), is in the order of 4,5K;

WebSplit AG18 Aharoni & Goldberg (Aharoni and Goldberg, 2018) arguing they could achieve more robust results from their split-and-rephrase models, proposed a new train-development-test data split corpus. They randomly divided the distinct complex sentences from the original WebSplit corpus across the TDT sets to ensure that every possible RDF relation is represented in the training set, and every RDF triplet is conferred in only one of the splits;

WikiSplit Botha and colleagues (Botha et al., 2018) introduced this corpus presenting a language-agnostic method for extracting split-and-rephrase rewrites from Wikipedia edit histories. Each single complex sentence maps to a single simplified reference containing only one split. Compared to WebSplit versions, this data set has a more rich and varied vocabulary over naturally expressed sentences, despite being slightly noisy. The authors showed that models trained on this data set produced dramatically better results;
## MinWikiSplit
This corpus is composed of 203K sentences whose referred simplified references are composed of shorter, syntactically simplified counterparts. As they specify, these are clauses with a 'minimal semantic unit that cannot be further decomposed into meaningful propositions' (Niklaus et al., 2019b). For this reason, the main contribution of this corpus is to possibly enable models to learn to perform more than one single split per complex input sentence. The authors did not state any division in train-development-test sets;

## PorSimplesSent
This is a corpus for sentence-based readability assessment in Portuguese. It is constructed from the PorSimples text simplification corpus (Caseli et al., 2009) and combines three levels of simplifications: from Original to Natural; from Natural to Strong; and from Original to Strong pairs (Leal et al., 2018). In this work, we employed the specific version of from Natural to Strong pairs that, in our view, better reflects a split-and-rephrase corpus. We selected only pairs with splits in the simplified side of the alignments, extracting 719 sentence pairs to test the pipeline in Brazilian Portuguese language.

For training purposes, we used both WikiSplit and MinWikiSplit training sets as they contain more rich and varied vocabulary with diverse syntax (see Section 4). The validations throughout implementation were performed using WebSplit v0.1, WebSplit AG18, and WikiSplit development sets. At last the results were obtained from WebSplit v0.1, WebSplit AG18, WikiSplit test sets, and PorSimplesSent (see Section 4.1). Table 1 summarizes the number of involved alignments from each corpus/set considering distinct complex sentences.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Training set</th>
<th>Dev. set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSplit v0.1 (Narayan et al., 2017)</td>
<td>-</td>
<td>554</td>
<td>554</td>
</tr>
<tr>
<td>WebSplit AG18 (Aharoni and Goldberg, 2018)</td>
<td>-</td>
<td>535</td>
<td>503</td>
</tr>
<tr>
<td>WikiSplit (Botha et al., 2018)</td>
<td>989,944</td>
<td>5,000</td>
<td>5,000</td>
</tr>
<tr>
<td>MinWikiSplit (Niklaus et al., 2019b)</td>
<td>203,309</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PorSimplesSent (Leal et al., 2018)</td>
<td>-</td>
<td>-</td>
<td>719</td>
</tr>
</tbody>
</table>

Table 1: Number of involved alignments in this work considering distinct complex sentences.

### 4 Experiments
We assembled two different training setups concerning the sequence-to-sequence neural models attending different corpora as follows.

#### Training Setup 1
From WikiSplit (Botha et al., 2018) training corpus, we selected aligned sentence pairs formed only by alphanumerical characters, commas, periods and whitespaces, eliminating any foreign/special characters as this corpus is slightly noisy as admitted by the authors\(^2\). This cut extracted 485,120 alignments, consolidating the training set for this first setup. We then executed our aforementioned custom implementation to construct the symbolic vocabulary and obtained 247 different items to train the first model;

#### Training Setup 2
From MinWikiSplit (Niklaus et al., 2019b) corpus, we first established a limit to select aligned sentences with a maximum length of 100 tokens, due to the fact that this corpus has few long sentences that would lead to long padding. This first cut extracted 197,496 alignments. We then repeated the prior setup selecting aligned sentence pairs formed only by alphanumerical characters, commas, periods and whitespaces, finally consolidating a training set of 122,104 alignments. The symbolic vocabulary obtained by our custom implementation was composed of 230 different items to train the second model.

#### 4.1 Results
Following Narayan et al. (Narayan et al., 2017), Aharoni and Goldberg (Aharoni and Goldberg, 2018) and Botha et al. (Botha et al., 2018) reference works, we report the results in sentence-level through BLEU (Papineni et al., 2002), BiLingual Evaluation Understudy, which is a primarily known metric borrowed from machine translation. It calculates modified \(n\)-gram precision as follows:

\[^2\text{Despite this training selection, the final predicted sentences by the pipeline can normally still have special characters assigned by the reconversion process.}\]
count the maximum number of times that an n-gram occurs in any of the references; (ii) clip the total count of each candidate n-gram by its maximum reference count; and (iii) add these clipped counts up, and divide by the total (unclipped) number of candidate words (Alva-Manchego et al., 2020). Also following reference works, we report the average number of simplified output sentences per complex input sentence (#S/C); and the average number of tokens per simplified output sentence (#T/S). Lastly, following Niklaus et al. (Niklaus et al., 2019b) we report the percentage of simplified output sentences that were totally copied from the complex input sentence without any modification (%SAME).3

Table 2 reports the obtained results against the full test sets when performing the complete pipeline with both trained models, considering the aforementioned different setups. Our best BLEU score was obtained with the model built by the Training setup 1 in the WikiSplit test set, closely followed by the score in the PorSimplesSent data. We also highlight the #S/C and #T/S features obtained with model from Training setup 2, pointing that this fashion attempts to split complex input sentences into shorter ones than the model from Training setup 1. The %SAME column values in turn illustrate our proposal’s low conservatism, tending to virtually intercept all the input sentences to perform the split-and-rephrase rewriting transformations (see detailed discussion in Section 5).

As expressed by the scores in Table 3 alongside other approaches scores (Copy512 and DisSim) in the WikiSplit test set, we established our pipeline as a competitive method. Copy512 is the strongest baseline reported by Aharoni and Goldberg (Aharoni and Goldberg, 2018) work. It is a sequence-to-sequence neural model augmented with a copy-mechanism (Gu et al., 2016) that bias the model towards copying tokens from the complex input sentences, taking into account that many of them should appear in the simplified output sentences. DisSim framework, by Niklaus et al. (Niklaus et al., 2019a), is a recursive sentence splitting approach, that applies a set of 35 hand-written rules to decompose a wide range of linguistic constructs, more oriented to generate simple and regular structures to support downstream semantic applications and faster generalization in machine learning tasks.

To encourage further research analysis, our complete logs containing all the predictions from Training setup 1 in the WikiSplit test set are publicly available4. One may notice many sentences achieved meaning preservation and perfect matches against the expected references.

5 Discussion

To achieve a detailed analysis, we manually inspected some of the predictions from the pipeline with the two built models bringing some examples to help explain the scores illustrated in Tables 2 and 3. These extracted examples are in Table 4 and show general patterns with some of the exciting behaviors produced by our method.

In Example 1, the same complex input sentence is transformed into different simplified outputs according to their training setups: Output 1 performed a single split whereas Output 2 performed two splits. This different behavior explains the higher numbers in the #S/C column and the lower numbers in the #T/S column from Training setup 2. These two measures confirmed the hypothesis that models trained in MinWikiSplit might capture the tendency to split source sentences into multiple output ones. Such multiple sentences may not be good for humans readers, but may benefit NLP downstream tasks.

In Example 2, even though both setups generated perfect outputs in terms of meaning preservation, only the Output 2 achieved maximum BLEU score since it is the unique that matched perfectly against one of the references. This brings the evidence that BLEU requires high-quality data to produce more precise outcomes, ideally with multiple correct references (Martin et al., 2019). Another limitation from BLEU is the low correlation with simplicity when sentence splitting is performed, but it still holds the high correlation with human assessments of grammaticality and meaning preservation (Alva-Manchego et al., 2020).

In Example 3, we note interesting contrasts produced by the models from the distinct training setups. While Output 1 retained the same structure from the complex input sentence, Output 2 promoted the reordering of the words preserving equivalent meaning and showing low conservatism. The only little mistake is observed by the repetition of word “was” in the Output 2.

3The metrics/quality estimation features were achieved with EASSE package (Alva-Manchego et al., 2019).

4https://github.com/pauloberlanga/split-and-rephrase-pipeline/
<table>
<thead>
<tr>
<th>Training setup 1</th>
<th>BLEU</th>
<th>#S/C</th>
<th>#T/S</th>
<th>%SAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSplit v0.1 (Test set)</td>
<td>58.34</td>
<td>2.17</td>
<td>12.52</td>
<td>0.014</td>
</tr>
<tr>
<td>WebSplit AG18 (Test set)</td>
<td>60.01</td>
<td>2.21</td>
<td>10.70</td>
<td>0.019</td>
</tr>
<tr>
<td>WikiSplit (Test set)</td>
<td><strong>68.92</strong></td>
<td>2.05</td>
<td>20.45</td>
<td>0.071</td>
</tr>
<tr>
<td>PorSimplesSent</td>
<td>65.00</td>
<td>2.06</td>
<td>14.95</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Training setup 2</th>
<th>BLEU</th>
<th>#S/C</th>
<th>#T/S</th>
<th>%SAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSplit v0.1 (Test set)</td>
<td>57.86</td>
<td>3.12</td>
<td>10.34</td>
<td>0.043</td>
</tr>
<tr>
<td>WebSplit AG18 (Test set)</td>
<td>58.45</td>
<td>3.17</td>
<td>9.03</td>
<td>0.033</td>
</tr>
<tr>
<td>WikiSplit (Test set)</td>
<td>44.65</td>
<td><strong>5.81</strong></td>
<td>11.78</td>
<td>0.011</td>
</tr>
<tr>
<td>PorSimplesSent</td>
<td>49.52</td>
<td>4.61</td>
<td>10.07</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2: Results obtained by the pipeline when applying both models built from the training setups.

<table>
<thead>
<tr>
<th>Models/pipelines</th>
<th>BLEU</th>
<th>#S/C</th>
<th>#T/S</th>
<th>%SAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training setup 1</td>
<td>68.92</td>
<td>2.05</td>
<td>20.45</td>
<td>0.07</td>
</tr>
<tr>
<td>Training setup 2</td>
<td>44.65</td>
<td><strong>5.81</strong></td>
<td>11.78</td>
<td><strong>0.01</strong></td>
</tr>
<tr>
<td>Copy512 (Aharoni and Goldberg, 2018)</td>
<td><strong>76.42</strong></td>
<td>2.08</td>
<td>16.55</td>
<td>13.30</td>
</tr>
<tr>
<td>DisSim (Niklaus et al., 2019a)</td>
<td>51.96</td>
<td>4.09</td>
<td>11.91</td>
<td>0.76</td>
</tr>
</tbody>
</table>

Table 3: Scores alongside other approaches in the WikiSplit test set.

Lastly, Example 4 illustrates the pipeline working in a cross-lingual manner. *Output 1* produced a pronoun “Ele” (He) instead of repeating “O projeto Gemini” (The Gemini project), as observed in *Output 2*. It is exactly the same behavior seen in the English Example 2 reflected for Brazilian Portuguese sentences. Recent studies that analyze eye movements of human readers interestingly reveal that they quickly retrieve information upon finding pronouns when referred to a close syntactic antecedent (Rebello et al., 2019).

Our detailed inspection together with the prediction logs confirmed that the pipeline could split complex input sentences into shorter simplified ones, often preserving equivalent meaning successfully. More than that, it showed ability to perform equivalent syntax transformations for different languages (English and Portuguese). On the other hand, some of the predictions reveal common mistakes from sequence-to-sequence models, such as repetition or omission of tokens and “hallucination” of new unwanted information. Another limiting factor is the noise from unsupported or missing statements observed in the referred test data sets. The low quality references eventually harmed the BLEU scores in those cases.

6 Conclusion

Split-and-rephrase task conceptualizes that shorter sentences are generally better processed by humans and by NLP downstream applications. We presented a complete pipeline for the split-and-rephrase method that attends in a cross-lingual manner English and Portuguese languages, by integrating sequence-to-sequence neural models and BERT’s masked language modeling. In contrast to conventional approaches, we train models making use of symbolic vocabularies defined by a custom implementation. This approach speeds up the training process and enables the models to acquire specific knowledge on how to split symbolic sentences, then demanding only a little step to convert them back to genuine texts in respective languages. Furthermore, the pipeline is capable to foster new words to rewrite the complex input sentence, thanks to BERT’s MLM predictions. Unlike most previous works on split-and-rephrase, we employed the four state-of-the-art corpora for the task and also a Brazilian Portuguese corpus, showing competitive results to equivalent approaches.

As future work, we plan to exploit our pipeline in more languages. We should also inspect the effectiveness of the Transformer architecture in replacement of the sequence-to-sequence models. Moreover, we intend to promote an extrinsic evaluation of the benefits of the split-and-rephrase method in NLP downstream applications.

---

3We refrain from report SARI and SAMSA scores. The first metric is more reliable to evaluate lexical (not structural) simplicity, and the second heavily relies on linguistic resources making the application in Portuguese language unfeasible.
**Example 1 (from WikiSplit data set)**

**Input**
Gavin confessed to the murder of George Pollard and was held in the Round House until he was hung on the 6th April 1844. His body was buried south of the Round House.

**Ref.**
Gavin confessed to the murder of George Pollard and was held in the Round House until he was hanged on 6 April 1844. His body was buried south of the Round House.

**Output 1**
Gavin confessed to the murder of George Pollard and was held in the Round House until he was hung on the 6th April 1844. His body was buried south of the Round House.

**Output 2**
Gavin confessed to the murder of George Pollard. Gavin was held in the Round House until he was hung on the 6th April 1844. His body was buried south of the Round House.

**Example 2 (from WebSplit v0.1 data set)**

**Input**
A.S. Livorno Calcio are managed by Christian Panucci who is attached to the club Genoa CFC.

**Ref.**
1. A.S. Livorno Calcio are managed by Christian Panucci. Christian Panucci is attached to the club Genoa CFC.
2. A.S. Livorno Calcio is managed by Christian Panucci. Christian Panucci played football for Genoa C.F.C.
4. A.S. Livorno Calcio is managed by Christian Panucci. Christian Panucci is attached to the club Genoa CFC.

**Output 1**
A.S. Livorno Calcio are managed by Christian Panucci. He is attached to the club Genoa CFC.

**Output 2**
A.S. Livorno Calcio are managed by Christian Panucci. Christian Panucci is attached to the club Genoa CFC.

**Example 3 (from WikiSplit data set)**

**Input**
Born in Huzhou, Zhejiang, Qian was trained in traditional Chinese philology, and was a student of Zhang Binglin.

**Ref.**
Born in Huzhou, Zhejiang, Qian was trained in traditional Chinese philology. He was a student of Zhang Binglin.

**Output 1**
Born in Huzhou, Zhejiang, Qian was trained in traditional Chinese philology and was a student of Zhang Binglin.

**Output 2**
Qian was born in Huzhou, Zhejiang. Qian was trained in traditional Chinese philology. Qian was a student of Zhang Binglin.

**Example 4 (from PorSimplesSent data set)**

**Input**
O projeto Gemini é resultado de uma associação de sete países e envolve a construção de dois telescópios com um espelho de oito metros de diâmetro.

**Ref.**
O projeto Gemini é resultado de uma associação de sete países. O projeto Gemini envolve a construção de dois telescópios com um espelho de oito metros de diâmetro.

**Output 1**
O projeto Gemini é resultado de uma associação de sete países e envolve a construção de dois telescópios com um espelho de oito metros de diâmetro.

**Output 2**
O projeto Gemini é resultado de uma associação de sete países. Ele envolve a construção de dois telescópios com um espelho de oito metros de diâmetro.

Table 4: Examples predicted by the pipeline with highlighted splitting points.
References


On the Contribution of Per-ICD Attention Mechanisms to Classify Health Records in Languages With Fewer Resources than English

Alberto Blanco¹, Sonja Remmer²,³, Alicia Pérez¹, Hercules Dalianis²,³, and Arantza Casillas¹

¹HiTZ Center - Ixa, University of the Basque Country UPV/EHU, Donostia, Spain
²Department of Computer and Systems Sciences, Stockholm University, Sweden
³Norwegian Centre for E-health Research, Tromsø, Norway

{alberto.blanco,alicia.perez,arantza.casillas}@ehu.eus
{remmer,hercules}@dsv.su.se

Abstract

We introduce a multi-label text classifier with per-label attention for the classification of Electronic Health Records according to the International Classification of Diseases. We apply the model on two Electronic Health Records datasets with Discharge Summaries in two languages with fewer resources than English, Spanish and Swedish. Our model leverages the BERT Multilingual model (specifically the Wikipedia, as the model have been trained with 104 languages, including Spanish and Swedish, with the largest Wikipedia dumps¹) to share the language modelling capabilities across the languages. With the per-label attention, the model can compute the relevance of each word from the EHR towards the prediction of each label. For the experimental framework, we apply 157 labels from Chapter XI – Diseases of the Digestive System of the ICD, which makes the attention especially important as the model has to discriminate between similar diseases.

1 Introduction

Electronic Health Records (EHRs) are classified by clinical experts for documentation, reporting global health vital statistics, insurance billing, etc. International Classification of Diseases (ICD) is used world-wide to define diagnostic terms and procedures and serves to encode EHRs. There are thousands of terms encoded within the ICD WHO (2016). For medical experts, reading EHRs, lengthy and technical documents, finding explicit and implicit mentions of diagnoses and procedures for then assigning standard ICD codes is cumbersome and requires specific training. In fact, it is well-known that manual encoding is not error-free, as an example, Jacobsson and Serdén (2013), estimated that 20% of them were either incorrect or were missing. In this context, natural language understanding brings opportunities to bridge the needs of the society in terms of computer aided coding approaches.

In 2006, it was argued that Natural Language Processing (NLP) tools could quickly help identify codes in discharge summaries Kukafka et al. (2006). Today NLP tools for classifying clinical documents written in English are widespread. Even more, languages with scarce resources for biomedical NLP like Spanish, Italian, Swedish, etc., are in the limelight in the last years to develop codification systems as has been done for English. In the context of working towards the codification of documents, in languages with a small number of resources for NLP, different tasks have been addressed. In 2018 CLEF Névéol et al. (2018b) worked with Italian, French and Hungarian for the automatic codification of death certificates. Each death certificate consisted of a few words (on average 20 words) with at least one main diagnosis. In 2020 the CodiEsp task at CLEF Miranda-Escalada et al. (2020) consisted on the automatic assignment of ICD-10 codes to Spanish Clinical Records with 350 tokens on average. For Swedish Henriksson et al. (2011) the authors mentioned that the corpus was compiled with documents that on average had a length of 96 words.

Admittedly, multi-label classification is challenging, particularly with extensive label-sets (as it is the case of the ICD) and domain-specific corpora, and even more when it comes to dealing with clinical information extraction on languages other than English Névéol et al. (2018a). Spanish and Swedish researchers are striving to bridge this gap, indeed, as the first and relevant step, they gathered corpora conveying patient records Oronoz et al. (2015); Dalianis (2018). Previous works showed that the multi-label classification problem of EHRs coded with ICD-10 can be tackled with an adapted

¹https://github.com/google-research/bert/blob/master/multilingual.md#list-of-languages
BERT architecture Amin et al. (2019); Zhang et al. (2020).

Moreover, we focused just on a sub-set of the ICD, i.e. the Diseases of the Digestive System (the ICD codes starting with the letter K). Focusing on semantically related diseases poses an added challenge, since the Natural Language Understanding (NLU) in charge of encoding the input EHR must be able to cope with the nuances inherent to the distinction of similar diseases. Unarguably, it is easier to distinguish two diseases each belonging to a different body-part than two diseases within the same body-part (as it is this case distinguishing diseases all within the digestive system). In summary, distinguishing semantically different diseases (e.g. gastrointestinal vs cardio-pulmonary) would be easier than distinguishing two diseases within the same speciality. To that end, the LM and the attention mechanisms play the most critical role, so we opted for the transformers models. BERT-based approaches have been tested in this context, with attention mechanisms as a strength towards finding relationships between input text with output ICD codes. The attention is a mechanism whose effectiveness has also been shown with other architectures such as RNNs with LSTM units Hochreiter and Schmidhuber (1997) or Convolutional Neural Networks Du et al. (2017).

Nevertheless, in this context we are dealing with scarce resources and relatively similar codes. In this line, the main scientific contribution of this paper rests on the implementation of a head adapted for BERT with multiple label attention mechanisms (instead of a generic one) in order to delve deeper into the nuances of the understanding module. In this work, we have implemented a per-label attention mechanism, and given that regular BERT models also have the self-attention mechanism, it allowed us to compare the effect of different attention mechanisms. The per-label attention mechanism allows the model to give a different relevance to each word and ICD code pair, contrary to the regular attention mechanism. The experimental results support the approach’s acceptable performance, so we decided to release the head for the scientific community.

2 Corpora

We have applied two datasets of languages with scarce resources for this work, i.e., languages with fewer resources than English, specifically, Spanish and Swedish. Both datasets are Electronic Health Records containing Discharge Summaries from patients. The Spanish EHRs are from the Emergency Services of the Basque Health Public System, conveying records, and therefore labels, from all the medical specialities Oronoz et al. (2015). However, the Swedish EHRs are only from the gastro-surgery medical specialisation and comes from the research infrastructure Health Bank - Swedish Health Record Research Bank², at Stockholm University. Therefore, to have equal label sets, we have selected the ICD codes shared between both datasets to carry out the experiments, obtaining 157 codes, all from the Chapter XI of the ICD-10, i.e., Diseases of the Digestive System. By selecting the codes of some specialities the number of available EHRs is reduced but the label sets are easier to handle. Training specific models on EHRs of specialities improves the performance against training general models Blanco et al. (2020). For the Swedish ICD-10 corpus data set the Swedish KB-BERT model Malmsten et al. (2020) has been applied with good results, see (Remmer et al., 2021).

Here we present a quantitative description and comparison between both datasets. Regarding the input, the Swedish dataset is more than twice larger in number of EHRs, with 8,909 records in contrast to the 3,891 available Spanish EHRs. Nevertheless, the vocabulary (i.e., number of unique words) is around three times bigger for the Spanish dataset. One explanation is that the Spanish EHRs come from several specialities, and therefore there is a higher lexical variability due to the specific terms of each medical specialisation. Also, the Spanish EHR contains lab tests, which could increase the number of unique words significantly.

Regarding the output, both datasets are equivalent, with the same set of 157 gastrointestinal ICD-10 codes. Although this is just a subset of the labels, there are still infrequent codes. For example, only 45 codes from the 157 appear in at least 1% of the EHRs. This fact makes the task even more challenging, as, for around 28% of the labels, there are only a few samples from where the model can learn. Even though the number of labels is the same, the distinct label sets (i.e., label combinations that are unique) are larger in the Swedish dataset than in the Spanish (1,288 and 558, respectively) due to the higher number of records. The ratio between the distinct label sets and the number of records

²http://dsv.su.se/healthbank
is similar, 6.97 for Spanish and 6.91 for Swedish, meaning that about the same number of EHRs lead to the same number of unique label sets.

The most significant differences come when evaluating the length of the EHRs, as the Spanish EHRs are significantly longer. While the Spanish records convey 984 words on average, the Swedish only have 74 words. The standard deviation is also more prominent in proportion, with 491 for the Spanish and 77 for the Swedish (note that the standard deviation is higher than the mean). Although the records from both datasets are Discharge Summaries, it seems that not all the Swedish records are complete summaries, but instead a summary or even one-sentence synopsis of the patient’s outcome.

3 Methodological Approach

Focusing the attention on the methodology, in Amin et al. (2019) the authors demonstrate the effectiveness of transfer learning with pre-trained language representation model BERT without attention for the multi-label classification of German non-technical summaries (NTSs) of animal experiments. In e-Health 2020 the authors of López-Garcia et al. (2020) tackled the task as a multi-label classification problem using BERT model Devlin et al. (2019) for the automatic clinical coding of medical cases in Spanish. NLU results crucial to this task and Transformers-based Language Models (LM) are, doubtlessly, the key strength of most recent approaches such as multi-label biomedical text classification Gu et al. (2020). All this and the inherent challenges related to our work (e.g. the ability to distinguish concepts leveraging semantically related diseases) motivated us towards BERT-based approaches. Another fact in favour to this choice rests on the ability to the transfer learning between the two languages and, if possible, get benefits from one Language Model to the other. That is, the resources from one language can boost the LM of the other one, while the system remains decoupled from the data.

In order to tackle the multi-label text classification task, we applied a model with a Transformer-based architecture. The problem to solve is the mapping between the input of the EHRs (the raw text, \( X \)) and a subset of ICDs from the entire label set, \( C \), where \(|C|\) is the number of codes. The Deep Learning model is trained for the downstream task with pairs of input and output (i.e., EHR texts and ICD codes). The Transformer-based neural network model is trained with instances comprising pairs of input (EHR text) and output (ICD codes). The transfer learning approach of sharing the LM of the other one, while the system boosts the LM of the other one, alleviates the training process for each language since just the task-dependent module (i.e. ICD multi-

From the input text, \( X_j \), fed to the Language Model part of the model, a hidden document representation is obtained. The importance of this rests in that our multi-label classifier is built on top of a BERT model (see Section 3.1). The LM is the core of the Transformer-based NLP models. The principal contribution of this work is the use of the hidden representation to compute attention weights that are label-specific for each input token. After computing the attention, the final output (label predictions) is computed with a fully connected layer that is fed with another document representation got from the label-specific attention layers. To support the reproducible research, we release the code of the per-label attention mechanism with this article.

3.1 Baseline: BERT to Boost LM

The Language Models based on Transformers, specifically BERT models Devlin et al. (2019), have been acknowledged due to their ability to generate contextual representations. In this work, we have to differentiate between very similar diagnoses (all from the gastrointestinal service), which motivated the chosen BERT model as the LM part of our multi-label text classification system to generate the representation of the EHRs. A BERT model is also suitable because of its built-in self-attention function, which can connect different locations of a single input sequence to one another. We also turned to BERT because it has been shown to expand Recurrent Neural Networks’ ability to model dependencies to long-distance patterns Hochreiter and Schmidhuber (1997).

In an attempt to encompass Spanish and Swedish, EHRs were represented with shared LMs. The transfer learning approach of sharing the LM poses two advantages. On the one hand, it alleviates the training process for each language since just the task-dependent module (i.e. ICD multi-
label classification) has to be trained. On the other hand, this bypasses the lack of in-domain data for languages other than English. Indeed, the multi-lingual LM, with English, leverages other languages such as Spanish and Swedish in a synergistic effect since cross-language regularities are captured by Pires et al. (2019).

The LM part is the core of the BERT models, but coupling different heads on top of the LM is what concedes the ability to tackle numerous downstream tasks, as multi-label classification. Since there are many parameters to describe both the LM and the head for the downstream task, training a BERT model is challenging. The LM module contains the broad majority of the parameters that must be inferred during the training stage. The ICD multi-label classification head built for this study, for example, accounts for less than 1% of the total model parameters (even though using the smallest variant of BERT, which has 110M parameters). With this in mind, we opted to train the multi-label heads from scratch while fine-tuning the LMs instead of training the LMs from scratch.

Because of memory and computational limitations, we used the BERTbase as the baseline BERT model (our GPUs are limited to 8GB of DRAM memory). The BERTbase model comprises 12 Transformers blocks, 12 self-attention heads, and an internal embedding layer size (d) of 768, totaling 110M parameters. The pre-trained BERTbase Multilingual model was used. The downstream tasks’ attention and output layers are connected to the output of LM, the hidden document representation, \( \mathbf{H} \), of the EHR.

### 3.2 Contribution: Per-ICD Attention Head

Having opted for the multilingual BERT to cope with the LM, next we proposed to improve the task-dependant head. The aim was to leverage ICD-dependant attention mechanisms in an attempt to enhance the model with added NLU capability when it comes to distinguishing ICDs within the same hospital-service (Digestive in our case).

Our multi-label classification head incorporates a per-label (per-ICD) attention mechanism. The model can classify the EHRs with respect to the ICD labels that are present through the text while also calculating the importance that each input token (word) has in relation to each of the ICDs.

Here, \( N \) is the number of tokens of the EHR (length) and \( d \) is the BERT hidden layer dimension (i.e., the representation of documents, being \( d = 768 \) for BERTbase models). Then, rather than perform the pool operation (across the document length, \( N \)), as in the original BERT Devlin et al. (2019) for classification, our head uses a per-ICD attention mechanism. The per-ICD attention mechanism allows the classifier to discover the correct relationships between the input tokens and each label.

For each ICD label, \( C_i \), the attention vector \( \alpha_{C_i} \in \mathbb{R}^{C_i \times N} \) is computed from the learnable vector parameter \( \mathbf{u}_{C_i} \in \mathbb{R}^d \), following (1), where \( C \) is the full set of ICD labels.

\[
\alpha_{C_i} = \text{Softmax}(\mathbf{H}^T \mathbf{u}_{C_i}) \tag{1}
\]

The attention scores must be computed as a probability distribution, representing the importance between each token and ICD label pair, and to that end, the model leverages the Softmax function. The matrix multiplication between \( \alpha \) and \( \mathbf{H} \) is calculated to get an ICD representation for each class from the attention weights. In the end, the maximum through the labels’ dimension is taken, obtaining the document representation on the final layer (\( \mathbf{v} \in \mathbb{R}^d \)), which combines the per-ICD attention representation.

The final layer of the head for multi-label classification is a regular one that allows getting the probabilities for each ICD label. It is a linear layer that takes the document representation (\( \mathbf{v} \)) as input, which takes into account the attention weights for each input token and label pair. After that, a Sigmoid function is applied to get the actual probabilities of each ICD, as in (2).

\[
\hat{y}_i = \sigma(\mathbf{W}_i^i \mathbf{v}_i + b_i) \tag{2}
\]

The probability of each ICD class (\( C_i \in C \)) being on the given input text is \( \hat{y}_i \). The parameters of the final layer are the weights matrix \( (\mathbf{W}) \) and bias \( (\mathbf{b}) \). Regarding the training of the model, it is carried out by minimising the loss function, precisely, the Binary Cross-Entropy (BCE) loss, as in (3). On this equation, the \( \hat{y} \) is the output of the previous final layer, and \( \mathbf{y} \) is the vector that encloses the ICD codes present on the EHR (i.e., the appearance or lack of ICD codes). Figure 1 shows an architectural outline of the system.

\[
\text{BCE}(\hat{y}, \mathbf{y}) = -W[\mathbf{y} \log(\hat{y}) + (1 - \mathbf{y}) \log(1 - \hat{y})] \tag{3}
\]
4 Experimental Framework

We propose the following experimental setup to evaluate our BERT model’s performance with per-ICD attention compared to the benchmark (standard BERT model) on the multi-label ICD classification downstream task. The experimental setup comprises the two minority languages (in terms of in-domain clinical data available), Spanish and Swedish, and a gastrointestinal label set of 157 labels. Each experiment is carried out twice, with the same experimental and training parameters, one with the regular multi-label classification head (as the baseline) and the other with our head with per-ICD attention. We show the results from the experimental results in Table 1 and Figures 2 and 3, for Spanish and Swedish, respectively.

The model with our per-ICD attention head obtains better results in both languages. It is important to note that the results improve considerably even in this context with a considerably large label set (157 labels). This finding is consistent with the following hypothesis: many terms can be important when dealing with a wide number of ICD codes at once and long EHRs, but probably only a few of them are relevant for each ICD code individually.

Multi-label ICD classification is often assessed by means of the Area Under the ROC Curve (AUC) micro averaging the metric for all the ICDs involved (denoted as AUCm in Table 1). For Spanish, the per-ICD model surpasses the base BERT model by 9.16 points, also improves slightly for the Swedish, with an improvement of around 1 point. In Figures 2 and 3 we show the confusion matrices for each experiment. Each confusion matrix is the average of the matrices of each ICD class, and we have computed two versions, i.e. one with arithmetic averaging (aka samples average) and the other with weighted averaging. In both, the darker the colour, the higher the metric, always in the range [0 – 100]. The weighted averaged matrices are computed considering the support (relative frequency) of each ICD class. Note that the TPR (True Positive Rate) and FNR (False Negative Rate) shown in Table 1 are also the arithmetic average of each corresponding model, but the CM show also the FPR (False Positive Rate) and TNR (True Negative Rate), while the weighted average of each metric. Regarding the per-class performance, there is a positive association with the support; the more frequent the label, the better are the results.

If we analyse the matrices, it can be observed that the source of improvement of the per-ICD model can be broken down; while the True Negatives stay close (as with a large label set, the majority of classes are negative), the True Positives improves considerably, with an increment of almost 100%. In the same way, the False Negatives decrease by around 20%. Although the Swedish results are in general weaker, this behaviour is appreciated similarly for both languages. Therefore, given the results, it seems that our per-ICD attention head is able to improve the Precision of the regular BERT models for ICD multi-label classification with large label sets. Nevertheless, the per-ICD model outperforms regular BERT in terms of performance, but also in interpretability capabilities, as it has the ability to export the attention weights, allowing its visualisation.

<table>
<thead>
<tr>
<th>L</th>
<th>Model</th>
<th>AUCm</th>
<th>TPR</th>
<th>FNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>baseline</td>
<td>58.16</td>
<td>17.70</td>
<td>99.21</td>
</tr>
<tr>
<td></td>
<td>per-ICD</td>
<td>67.32</td>
<td>34.92</td>
<td>99.38</td>
</tr>
<tr>
<td>SW</td>
<td>baseline</td>
<td>54.92</td>
<td>15.49</td>
<td>92.24</td>
</tr>
<tr>
<td></td>
<td>per-ICD</td>
<td>55.96</td>
<td>27.91</td>
<td>82.45</td>
</tr>
</tbody>
</table>

Table 1: Comparison of results on the Spanish (SP) and Swedish (SW) datasets (“L” stands for “Language”) obtained with the baseline BERT and BERT enhanced with per-ICD attention head. TPR is the True Positive Rate and FNR the False Negative Rate.

5 Discussion

Within the clinical text mining field, the main weakness tends to be the availability of corpora due to the natural patient’s confidentiality policy Cohen and Demner-Fushman (2014). As a result, for the research to make progress, the so important comparability might get compromised. By contrast, through this work the authors are glad to make available their own implementation of the per-ICD attention approach as a secondary contribution of

3To get the source code of the implementation, simply e-mail the first author.
Another aspect related with the corpus is the complexity and length of the input EHR. The average length of the input of the works mentioned Névéol et al. (2018b); Cappellato et al. (2019) are variable from a few words in the case of Italian, Hungarian and French to 350 words for the documents written in Spanish Miranda-Escalada et al. (2020). By contrast, in our paper we deal with documents in Spanish and Swedish with an average length of 800 (exceeding the aforementioned ones) and 70 respectively.

According to these results, the per-label attention mechanism improves Precision. While more performance is still necessary for a fully automated system, the results suggest that it is suitable for multi-label classification of EHRs according to the ICD standard, specifically applying it as a clinical DSS, as the per-ICD attention can aid the expert in the EHR codification process.

6 Conclusions

We have dealt with the codification of EHRs of the gastrointestinal service for Swedish and Spanish hospitals. We have developed a BERT model for multi-label classification incorporating a per-label attention mechanism.

The results obtained have revealed that the proposed model outperforms the regular BERT. We have proved this fact for two languages with minority resources in clinical NLP, showing that solutions of language independent nature work. Moreover our proposal generates an interpretable output that helps to know the relevance of the tokens with respect to each ICD assigned to the EHR. To sum up, the per-label attention mechanism differentiates semantically ICDs that are related and aids to explain the core of each label. Future work may include testing BERT models trained for the specific languages, as the BETO model Cañete et al. (2020) for Spanish.

Acknowledgments

This work was partially funded by the Spanish Ministry of Science and Innovation (DOTTHEALTH/PAT-MED PID2019-106942RB-C31), European Commission (FEDER) and the Basque Government (IXA IT-1343-19, Predoctoral Grant PRE-2019-1-0158) and by the ClinCode project, project number 318098, from the Norwegian Research Council. This research has been approved by the Regional Ethical Review Board in Stockholm under permission no. 2007/1625-31/5. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.
References


Can the Transformer Be Used as a Drop-in Replacement for RNNs in Text-Generating GANs?

Kevin Blin  
Section of Communication Systems  
Bachelor semester 6  
EPFL, Lausanne, Switzerland  
kevin.blin@epfl.ch

Andrei Kucharavy  
Distributed Computing Laboratory  
EPFL, Lausanne, Switzerland  
andrei.kucharavy@epfl.ch

Abstract

In this paper we address the problem of fine-tuned text generation with a limited computational budget. For that, we use a well-performing text generative adversarial network (GAN) architecture - Diversity-Promoting GAN (DPGAN), and attempted a drop-in replacement of the LSTM layer with a self-attention-based Transformer layer in order to leverage their efficiency. The resulting Self-Attention DPGAN (SADPGAN) was evaluated for performance, quality and diversity of generated text and stability. Computational experiments suggested that a transformer architecture is unable to drop-in replace the LSTM layer, under-performing during the pre-training phase and undergoing a complete mode collapse during the GAN tuning phase. Our results suggest that the transformer architecture need to be adapted before it can be used as a replacement for RNNs in text-generating GANs.

1 Introduction

Since the introduction of the Transformer in late 2017 by Vaswani et al. (2017), pure self-attention architectures have become ubiquitous in the natural language processing community.

Initially introduced by Bahdanau et al. (2015), the self-attention mechanism proved to be particularly useful in the context of neural machine translation to augment existing recurrent neural networks (RNNs). Recurrent neural networks (RNNs), introduced by Rumelhart et al. (1986) and expanded by Hochreiter and Schmidhuber (1997); Cho et al. (2014), were the state of the art in learning and generating sequential models, notably for texts. With the addition of the attention mechanism, in the context of machine translation, they could focus better on the words and word combinations corresponding to the same concepts in different languages and focus on learning equivalences between them as opposed to trying to infer them directly from the whole excerpts of parallel texts used for learning.

Following the initial introduction of the attention mechanism to augment the RNNs, architectures combining the two became de-facto state of the art. Stacking RNN layers, adding pass-through mechanisms and separating the architecture into encoder and decoder with an attention layer in the middle became a standard, powering among others Google's Neural Machine Translation system (Wu et al., 2016; Johnson et al., 2017). Despite impressive performance, those architectures had a fundamental limitation to their ability to scale. The RNN training and evaluation are sequential by their nature, which means that architectures relying on the RNNs could hardly benefit from the arrival of massively parallel computing.

The innovation of the Transformer was to show that it was possible to learn sequence-to-sequence mapping while dispensing entirely with RNN layers, using only self-attention mechanisms ("Attention is all you need") - Vaswani et al. (2017)). By stacking several layers of self-attention networks to form an encoder and a decoder, as well as introducing multi-head architectures, where each layer of self-attention network could be trained in parallel, the Transformer and architectures derived from it scaled up easily and could be trained in parallel at a scale that was previously unreachable.

That scalability enabled a continuous ramp-up of performance through parameter and training dataset size increase (Brown et al., 2020), which eventually hurt itself against the limitations of reasonable demands on computational power in deployments. In turn, this led to an extensive research into making Transformer-based architectures more efficient, focusing at first on specific instances (Jiao et al., 2020; Sanh et al., 2019) and more recently on more general approaches (eg. Li et al. (2020);
Mandava et al. (2020); Fedus et al. (2021); Ren et al. (2021)). Combined with the existence of specialized, energy-efficient hardware that is highly compatible with the Transformer architectures, this makes Transformer-based architectures an attractive architecture to get the best value of limited computational budget.

The text-generating capabilities of the Transformer also gave rise to a new generation of models specialized in text generation. Rather than mapping texts between languages, they focused on mapping a prompt to a text that would follow it. Google’s BERT model (Devlin et al., 2019) isolated and scaled up the Transformer encoder stack in order to perform masked training - predicting masked words in a text - referred to as autoencoding models (Rumelhart et al., 1986; Hinton et al., 2006; Erhan et al., 2010). On the other side of the spectrum, OpenAI’s generative pre-training (GPT) family of models (Radford et al., 2018, 2019; Brown et al., 2020), focused on approximating the decoder stack and using the tokens from the prompt in order to initialize the hidden state of the self-attention modules stack - referred to as autoregressive models (by similarity with statistical autoregressive methods (Yule, 1926; Wold, 1938; Slutzky, 1937; Box and Jenkins, 1970), see Bowman et al. (2016) for machine learning applications).

Despite their impressive performance, when it comes to the text generation, both types of models are essentially autoregressive and are trained by max likelihood methods with regards to the training datasets. This poses several challenges. First, the autoregressive models learn the next token as a continuation of real texts they encounter in training, yet during the generation phase they continue from the text they themselves have generated. This means that during the generation phase they can rapidly go off the deeper end into an uncharted territory, they don’t have a statistical model, and start generating degenerate output - a problem referred to as exposure bias (Holtzman et al., 2020). So far, solutions to this problem, such as scheduled sampling (Bengio et al., 2015), are far from perfect and result in less diverse sampling and mode collapse (Huszar, 2015). On top of that, the autoregressive nature of the model means that even in the territory where it has learned an appropriate token distribution, it will still be learning potentially undesirable biases (Hutchinson et al., 2020), with no means to correct them other than to curate the entire dataset used to train the model (reviewed in depth by Bender et al., 2021).

Trying to learn the explicit statistical structure of natural language is, however, not the only way to train generative models. Adversarial Generative Networks (GANs) are a different training mode, where a generative model learns to generate outputs that are indistinguishable from the ones in the training dataset through a competition with a critic model, trained in tandem with it. Introduced by (Goodfellow et al., 2014), they are more robust to output degeneration, given that they always train in the generative mode, and require less computational resources than traditional autoregressive models. Besides, a number of different pre-trained critics can be used to eliminate undesired biases or on the contrary, introduce desired ones, such as specifying an artistic style of an image (Gatys et al., 2015).

The adversarial learning approach has been highly successful for training image generation models, allowing high-quality image generation (Brock et al., 2019), day-to-night or summer-to-winter image translation (Isola et al., 2017), or sketch-to-image translation (Lu et al., 2018). GANs application to text generation, however, remained relatively limited. A major reason for that is that the sampling step leading to discrete and sequential token organization, needed for text generation, is problematic for gradient estimation, which is essential for training GANs. As such, most existing text-generating GANs rely on a max-likelihood autoregressive pre-training with the actual adversarial training phase being short and using a small learning rate, similar to a fine-tuning. Unfortunately such an approach failed to address the shortcomings of the purely autoregressive models they acquire during the pre-training phase.

However, recently de Masson d’Autume et al. (2019) was able to demonstrate that it was possible to train a text-generating GAN architecture from scratch, thus avoiding entirely the problems encountered by autoregressive max-likelihood methods. A curious property of the Scratch-GAN he developed, is that while it seems to solve problems that plagues both RNNs and pure self-attention autoregressive models learning the explicit text token distribution, the authors still opted for RNN blocks, foregoing the advantages of Transformer architectures in NLP and self-attention in image-generating GANs (Zhang et al., 2019).
Given the massive advantages of the Transformer presents when it comes to training as well as with regards to the amount of research performed to make them more efficient, as well as better capabilities of Transformers compared to GANs, we wanted to know if it was possible to perform a drop-in replacement of RNNs with Transformers. Such a Transformer-based GAN could provide two main benefits: the ability of produce higher quality samples at a reduced computational cost compared to traditional RNN-based GANs, and a more scalable language GAN.

In order to approach this question, we chose to perform an experimental evaluation, based on a classical text-generating GAN - Diversity Promoting GAN (DPGAN), developed by Xu et al. (2018). We chose DPGAN due to its straight-forwards architecture and training mode, similarity of its rewards structure to the state of the art text-generating GANs de Masson d’Autume et al. (2019) and the presence of the maximum likelihood pre-training, that we were expecting to be particularly favorable to the Transformer layers. We refer to the DPGAN with RNN layers replaced by a transformer as Self-attention DPGAN (SADPGAN).

2 Related Work
The power of the Transformer architecture in the language modelling tasks and its potential to further improve existing GAN architectures did not escape the attention of the machine learning community. So far, self-attention architectures in GANs focused on image generation tasks. Perhaps the two best known examples are TransGAN (Jiang et al., 2021) and Self-Attention GAN (SAGAN) (Zhang et al., 2019). More recent advances, such as the introduction of Generative Adversarial Transformers by Hudson and Zitnick (2021) have build on the Transformer architecture even further, enabling long-range correlation to improve over the existing state of art, showing a potential for it in the GAN setting. However, a common point between all of them is that they focus on the generation of images, rather than texts, and use only a single encoding layer for self attention or attempt to modify self-attention mechanism to better suite the image generation, diverging from the Transformer architecture.

The approach to text generation by combining the Transformer and GANs that comes closest to ours is the SALSA-Text, developed by Gagnon-Marchand et al. (2019). SALSA-Text is a text-generating GAN build around a Transformer, discarding the original layer normalization and replacing it with the spectral layer normalization. In addition to that, SALSA-Text uses a modified Transformer architecture, with less layers and a different structure, as well as a specific training regimen, meaning it is more of a GAN built around a Transformer rather than a GAN where an RNN layer has been replaced with a pure self-attention based layer.

Here, we examine the Transformer applicability as a general-purpose element that can be drop-in in architectures requiring a latent space encoding and thus directly replace RNNs structures as LSTMs and Gated Recurrent Units (GRUs).

3 Contribution and Outline
Our contribution consisted in assessing the potential of Transformers as a drop-in replacement of LSTMs in text-generating GANs. Through three different experiment involving SADPGAN, we showed that Transformers, despite achieving remarkable results in several NLP tasks, fail to adapt to the adversarial learning and adversarial fine-tuning context (Jeddi et al., 2020), causing SADPGAN to consistently come short of the DPGAN performance and to exhibit severe mode collapse. This results suggest that as of now, the Transformer can’t be directly used as a replacement for LSTM without further architectural and training mode changes.

4 Methodology
Our work is built upon an existing PyTorch implementation of DPGAN by Liu et al. (2020b).

We used an iterative approach, first implementing a layer using exclusively Transformer encoders, then adding the decoder stack with masking and finally adding teacher forcing during training. For each of these steps we compared the training results to the original implementation.

Since our goal was to investigate the possibility of replacing LSTM layers in text-generating GANs by dropping in Transformer layers, and not to achieve new state of the art for text-generating GANs, we kept our models relatively small: embedding and hidden dimension of 32, Transformer encoder/decoder with two layers, each with 4 attention heads of size 64.

The training loop for both GANs consisted in
120 iterations of MLE pre-training for the generator followed by another 120 epochs of adversarial training between the discriminator and generator. To assess the performance of both architectures, we used two negative log likelihood metrics, $NLL_{gen}$ and $NLL_{div}$, which measure the quality and respectively diversity of the generated text and (self)BLEU scores when using real data. For the NLL metrics lower values correspond to better results while for BLEU higher scores are desirable. The code developed for this project is available from https://github.com/TheBlueHawk/RANLP21-70, specifically the RANLP-2021 release. For the ease of use, it was packaged integrated into the code from Liu et al. (2020b).

5 Results and Discussion

5.1 Experiment #1

Following the example set by Jiang et al. (2021) in image GANs, we first tried to drop-in a Transformer encoder block, ensuring that it would properly fit the rest of the GAN architecture. For this preliminary experiment we trained both SADPGAN and DPGAN on a synthetic dataset with a vocabulary of 5000 words.

As shown in figure 1, for SADPGAN we didn’t observe any improvement during MLE training. This can be explained by the choice of architecture: using only Transformer encoders without combining them with up-sampling layers for image generation or using bidirectional attention heads as in Devlin et al. (2019), doesn’t allow the model to correctly learn how to produce realistic samples.

5.2 Experiment #2

Following the results of the first experiment, we modified the Transformer block by adding a Transformer decoder after the Transformer encoder. Contrary to an encoder-only architecture, the addition of the decoder requires a second input, the output vector (also known as target), which is fed to the first masked multi-head attention layer of the decoder. To leave a maximum freedom to the decoder when generating sentences, we decided to feed it with an empty vector. For this second experiment we trained both model on real data, specifically the Lin et al. (2014) annotation dataset using a pre-trained Word2Vec embedding. Word samples were obtained in parallel using multinomial sampling.

The set of modifications permitted SADPGAN to improve the quality of the generated text during the MLE pre-training iterations but the learning curve flattened out rather quickly and performed worse than the original implementation, with also a decrease of output diversity. During the adversarial training we observed a severe mode collapse: the original DPGAN produced very similar sentences while SADPGAN eventually produced exclusively empty sentences. See table 1 for some samples randomly drawn at the end of pre-training and adversarial training.

The reason for this particular behaviour lies in the role of the target vector, which act as a ground truth in a teacher forcing scenario. This means that parallel sampling cannot be performed without forcing the use of the empty sentence as a reference.

5.3 Experiment #3

To address the problems of experiment #2 we used two modifications: a new target vector and autoregressive sampling. As a target vector we used the input sentence to the encoder but shifted right by one position: this allowed a correct implementation of teacher forcing during the training phase. During the evaluation phase or when sampling, we don’t want the teacher to influence the output of
Table 1: Text samples produced after pre-training (pre) and adversarial training (adv) by GAN architectures

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Pre-training Sample</th>
<th>Adversarial Training Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPGAN</td>
<td>a person sits on black motorcycle on a busy bench near the side</td>
<td>a man riding a motorcycle down a street</td>
</tr>
<tr>
<td>SADPGAN</td>
<td>flies a various a trash a while begs area in refrigerator</td>
<td>&quot;empty sentence&quot;</td>
</tr>
</tbody>
</table>

Figure 2: $NLL_{div}$ and $NLL_{gen}$ losses of SADPGAN and DPGAN during the second experiment.

Figure 3: $NLL_{div}$ and $NLL_{gen}$ losses of SADPGAN and DPGAN during the third experiment.

The results of pre-training show a greater improvement of the quality of samples but at the cost of severe lack of diversity, resulting in common words repeated in sequence. Again, the adversarial training further exacerbate the issues resulting in total mode collapse. BLEU and self-BLEU scores for different n-grams at the end of pre-training and adversarial training are reported in table 2 for both DPGAN and SADPGAN.

6 Conclusion

Following our results, we have observed that a Transformer architecture cannot be used as a simple drop-in replacement for RNNs in the context of text-generating GANs, at least not in its unmodified form.

This result is not entirely surprising. Despite a great performance of the Transformer and its derivatives in different NLP tasks, Transformer-based architectures are all but simple to train, as reviewed in Liu et al. (2020a).

Another problem with Transformer architectures is their tendency to overfit the distribution of the text they are learning and fail to generate novel text in case insufficiently diverse and varied training datasets are used. In the case of the DPGAN, the small size of the training dataset and the relatively small size of batches likely place us directly into the overfitting territory. Upon the transition into the adversarial training, this overfitting is likely mutually amplified, leading to more and more degenerate text outputs.
A more fundamental problem is that in DPGAN, we are still faced with the transition from the max-likelihood training regime to a generative regime. Which means that even with improved datasets, the generator is likely to wander off into the uncharted territory as it tries to generate a new token based on the tokens it has already generated rather than tokens sampled from the training dataset. While this problem is present as well with the RNNs, it seems that the output quality decay occurs faster with the Transformer-based architecture.

Despite the existing wealth of text-generating GANs, none, except for SALSA-text (Gagnon-Marchand et al., 2019) are Transformer-based, but persistently use RNNs, notably LSTM layers, including in the most recent, state-of-the-art ones, such as (de Masson d’Autume et al., 2019). In case of SALSA-text, a modified architecture and a different regularization methods are used, both in two specific setups. It seems that part of this trend in using RNNs rather than a Transformer-based architecture is rooted in the existence of fundamental differences in the way the Transformer learns compared to the RNNs, that make it particularly vulnerable to the hazards of adversarial training regiments.

As a result, we expect that developing text GAN architectures using self-attention based architectures instead of RNNs ones would require designing new GAN architecture from scratch to ensure a initialization and evaluation/reward structure that would be compatible with the the Transformer layer. A pure self-attention layer capable to be a drop-in replacement for RNNs is still to be developed. Given the similarity between the adversarial phase occurring in the majority of current text-generating GANs and the adversarial fine-tuning mechanism (reviewed by eg. Jeddi et al. (2020)), we expect that text-generating model fine-tuning to avoid undesirable patterns and adversarial prompts (such as exemplified in Bender et al. (2021)) would not be straightforwards and rely on the development of more robust and stable self-attention architectures.

7 Future Work

While this paper presents a negative result, we did not evaluate a number of approaches that could stabilize the training of pure self-attention architectures. Such approaches could prove to be key to the development of self-attention architectures that can be used as drop-in RNN layers replacements.

An interesting avenue is to build upon the achievements of Gagnon-Marchand et al. (2019), and start with a reduced Transformer architecture, combined with a spectral normalization. Building up on this idea, Noise Stability Regularization proposed by Hua et al. (2021), suggested that noise regularization methods were capable to significantly improve the stability pre-trained pure self-attention generative networks fine-tuning, which could indicate an overall improvement in stability that would be visible in generative adversarial training. The regularization approach has been as well highlighted by several other publications, such as Nguyen and Salazar (2019) and would be the first line of research to be investigated.

Another angle of attack would be to increase the amount of tokens used for initialization of the Transformer on the generator side as well as to perform the initialization on multiple levels. This approach have been shown to perform well in Shin et al. (2020). Similarly, existing Transformer architectures are known to require learning rate schedulers for training, something that seems to be entirely absent from almost all existing text-generating GAN architectures. It is possible that

<table>
<thead>
<tr>
<th>n-gram BLEU Scores</th>
<th>Self-BLEU Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DPGAN</strong></td>
<td></td>
</tr>
<tr>
<td>pre</td>
<td>0.73 0.497 0.306 0.185</td>
</tr>
<tr>
<td>adv</td>
<td>0.845 0.737 0.578 0.455</td>
</tr>
<tr>
<td><strong>SADPGAN</strong></td>
<td></td>
</tr>
<tr>
<td>pre</td>
<td>0.276 0.065 0.03 0.019</td>
</tr>
<tr>
<td>adv</td>
<td>nan</td>
</tr>
</tbody>
</table>
a learning rate scheduler needs to be incorporated into a Transformer layer for it to be able to become generally applicable. The importance of the learning rate scheduler for Transformers has been extensively documented in the past, notably by Popel and Bojar (2018) and could also be key in stabilizing the training of Transformer-based architectures.

Overall, in the absence of proof that pure self-attention architectures are inherently unstable in the adversarial training context, there is a number of potential approaches to make them work in the context of GANs and leverage their advantages that are to be explored.

Acknowledgments

We would like to thank the Armasuisse - Swiss Cyber Defence Campus for the Distinguished Post Doctoral Fellowship supporting AK, and for providing the computational infrastructure for experiments, as well as Drs Rachid Guerraoui (EPFL) and Ljiljana Dolamic (CYD/Armasuisse), for oversight and general advice on this project, and finally Fabien Salvi (EPFL) for the technical support regarding the computational infrastructure organization.

References


Predicting the Factuality of Reporting of News Media
Using Observations About User Attention in Their YouTube Channels

Krasimira Bozhanova  
FMI, Sofia University  
“St. Kliment Ohridski”  
Sofia, Bulgaria  
krasimira.bozhanova@gmail.com

Yoan Dinkov  
FMI, Sofia University  
“St. Kliment Ohridski”  
Sofia, Bulgaria  
oyandinkov@gmail.com

Ivan Koychev  
FMI and GATE, Sofia University  
“St. Kliment Ohridski”  
Sofia, Bulgaria  
koychev@fmi.uni-sofia.bg

Maria Castaldo  
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, GIPSA-lab, F-38000, France  
maria.castaldo@ grenoble-inp.fr

Tommaso Venturini  
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, GIPSA-lab, F-38000, France  
tommaso.venturini@cnrs.fr

Preslav Nakov  
Qatar Computing Research Institute, HBKU  
Doha, Qatar  
pnakov@hbku.edu.qa

Abstract

We propose a novel framework for predicting the factuality of reporting of news media outlets by studying the user attention cycles in their YouTube channels. In particular, we design a rich set of features derived from the temporal evolution of the number of views, likes, dislikes, and comments for a video, which we then aggregate to the channel level. We develop and release a dataset for the task, containing observations of user attention on YouTube channels for 489 news media. Our experiments demonstrate both complementarity and sizable improvements over state-of-the-art textual representations.

1 Introduction

Disinformation in the news and in social media is perceived as having a major impact on society, e.g., during the 2016 US Presidential election (Grinberg et al., 2019) and the Brexit referendum (Gorrell et al., 2018). During the COVID-19 pandemic outbreak, the moral panic (McLuhan, 1964) around online disinformation grew to a whole new level as the first global infodemic.¹ Nowadays, fighting disinformation online grew to a whole new level as the first global infodemic.¹ Nowadays, fighting disinformation online has been recognized as one of the most important issues societies around the world are facing today.

In this paper, we highlight an aspect of disinformation that is often neglected. Rather than examining the truth-value of individual piece of information, we investigate the general quality of the attention regimes in different news outlets, by analyzing news media’s YouTube channels.

YouTube is the largest and the most popular platform for video sharing with over two billion users and it is also the second most widely used news source in USA, after Facebook.² While the platform has been scrutinized for the way in which it may amplify marginal and sometimes radical contents (Munn, 2019; Ribeiro et al., 2020), the connection between attention dynamics and disinformation levels is still largely unexplored.

Here, we do not focus on specific videos, but rather on entire YouTube channels of news media and their attention dynamics. In particular, we are interested in differentiating the “attention cycles” (Downs, 1972; Leskovec et al., 2009) of YouTube channels, that is to assess the rapidity and the steepness with which their videos rise and fall in the consideration of their audiences. While some outlets encourage extensive and diverse discussions, other tend to concentrate everyone’s attention on the latest hot-button, thus distracting the public opinion instead of nourishing it (Venturini, 2019).

Our contributions are the following:

- We propose to model the factuality of news media based on the user attention cycles in their respective YouTube channels.
- We release a specialized dataset for the task.³
- We show experimentally that considering attention cycles yields considerable performance gains on top of text representations for predicting the factuality of news media.

¹MIT Technology Review: tinyurl.com/y8oschng
²http://tinyurl.com/y4apu58j
³http://github.com/krasimira-bozhanova/youtube-attention-cycles-dataset

187
The paper is organized as follows: Section 2 presents related work. Section 3 describes our dataset. Section 4 discusses our methodology. Section 5 presents the experiments and results. Section 6 offers analysis and discussion. Section 7 concludes and points to directions for future work.

2 Related Work

Significant efforts have been dedicated in the last years to automating the detection of disinformation (which is commonly referred to as fake news), which we look at more closely in this section. At the end, we mention previous attempts to use data from YouTube for media classification tasks.

Analysis of the Content Many approaches have been proposed to analyze both the style and the content of fake news. Using natural language processing techniques, Horne and Adali (2017) pointed out how fake news can be characterized by stylistic features such as the overuse of proper nouns, punctuation, capital letters, negation terms, and repetitions in the text (Rubin et al., 2016). Fake news has also been associated with intensity of sentiment and emotions, compared to mainstream news (Gianchanou et al., 2019). Here, we also use textual representations, but (i) we focus mainly on analyzing the user attention cycles in YouTube channels, and (ii) we aim at categorizing entire news media outlets rather than individual pieces of news.

Analysis of the Response Many researchers used the user reactions in social media platforms to identify disinformation, e.g., the content and the number of replies to a piece of news, or the propagation of the content in the network. For instance, Zhao et al. (2015) classified disputed claims based on their comments and reactions, assuming that, if a claim is not true, at least some replies would question its factuality. Indeed, later studies (Ruchansky et al., 2017; Nguyen et al., 2020) have shown that user response features are quite important. Here, we also focus on inspecting the user response and its potential link to disinformation, but we use user attention cycles in YouTube.

Analysis of Entire News Media Outlets Looking at a news media outlet as a source of low-quality content is another way to approach the problem. In these methods, features modelling the overall trustworthiness of the source are used, such as (i) Does the news media outlet use verified accounts on established platforms, such as Wikipedia and Twitter? (ii) If it does, do these accounts have a proper description, location, website references, etc? (iii) How does the URL of the media’s website look like? (iv) Does the medium express political bias or sentiment? Baly et al. (2018) and Baly et al. (2020) used features motivated by these questions to achieve better results in combination with content features and user profiles in social media. In our work, we also aim at classifying entire news media outlets, but we do so using user attention cycles in YouTube along with text.

Using Temporal Attention Data for Disinformation Detection We focus on the analysis of temporal patterns associated with news outlets of different type and quality. Previous studies (Ruchansky et al., 2017; Nguyen et al., 2020) have suggested that a combination of temporal, content-based, and user-based features is promising for disinformation detection. As the dynamics of the viral spread is often associated with successful junk news, we look at studies focusing on modelling virality, such as (Hoang et al., 2011) for tweets. Similar features are well-suited for our task, as described in Section 4.1.2, but (i) we model the user behavior differently, and (ii) we focus on data collected from the YouTube channels of the target news media.

Using YouTube Data for Classification The YouTube platform contains information that is still underexplored for the purposes of disinformation detection. Dinkov et al. (2019) looked into detecting the left-centre-right political bias of YouTube channels. Baly et al. (2020) included features from the news source’s YouTube channel, derived from both sound and user profiles. The above work uses raw statistics about the number of views, likes, dislikes, and comments per video. We, instead, use much richer temporality features in combination with the textual representation of the videos.

3 Data

We started from a corpus of news media outlets, whose reliability has been evaluated by Media Bias/Fact Check (MBFC). Lead by a team of independent journalists and researchers, MBFC has analyzed close to 4,000 news outlets over the past six years. For each news outlet, they provide a detailed analysis summarized by a ‘factuality’ score chosen among: Very High, High, Mostly Factual, Mixed, Low, and Very Low.
Factuality | Channels | Videos |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>308</td>
<td>22,932</td>
</tr>
<tr>
<td>Mixed</td>
<td>153</td>
<td>12,125</td>
</tr>
<tr>
<td>Low</td>
<td>28</td>
<td>2,091</td>
</tr>
<tr>
<td>Total</td>
<td>489</td>
<td>37,148</td>
</tr>
</tbody>
</table>

Table 1: Statistics about the dataset, showing the distribution of the channels and of the videos for each level of factuality of reporting.

We searched the YouTube channels of news outlets in the MBFC corpus and we monitored all the videos they published from February’2020 to August’2020. Using the YouTube Data API,\(^5\) we collected the number of views, likes, dislikes and comments collected during the first seven days after the publication of each video. We also stored its title and its description.

We observed that the percentage of media channels labelled with the Very High and the Very Low categories was 3.1% and 1%, respectively. We thus merged Very High with Mostly Factual with Mixed; and Very Low with Low, ending up with a 3-way labelling: High, Mixed, and Low.

Finally, for the sake of data balancing, we excluded the channels with fewer than 20 videos, and we capped the most prolific channels at the newest 100 videos. The final distribution of channels and their corresponding videos for each level of factuality is shown in Table 1.

4 Method

Our system is composed of two main components focusing on (i) data preparation, and on (ii) sequential classification, respectively. Below, we describe the representation we use for video-level and also for channel-level classification.

4.1 Representation

The data preparation component transforms the YouTube source data and produces representations (or features) for our model. We generate representations both for the textual content of the videos and for the user attention data, for which we introduce a number of novel features, presented in Section 4.1.2.

4.1.1 Textual Features

We gathered the title and the description of each video. We extracted Sentence BERT embeddings (768 features) for each title and description. These embeddings are derived from a modification of the pretrained BERT model, which yields semantically meaningful sentence embeddings of size 768, which are trained to be readily comparable using cosine similarity (Reimers and Gurevych, 2019).

4.1.2 Attention Features

We hypothesize that the attention received over time by the videos in a YouTube channel can be used to predict the quality of its contribution to the online public debate, as captured (albeit imprecisely) by the factuality score assigned by MBFC. We generate a set of features that model the user attention cycles by looking at the temporal variation of the number of user actions (Views, Likes, Dislikes, and Comments) in the first week after a video has been published. We aim to model the following:

- How are user actions distributed hourly/daily?
- How much are the user actions concentrated in peak hours?
- At what moment in time does the peak hour for each user action type occur?
- How steep is the time series in terms of the distribution of hourly user actions?

We define \( UA_{d_i} \) as the total number of user actions that occurred by the end of day \( i \). In our case, \( i \) ranges in \( \{1, 2, \ldots, 7\} \). Similarly, \( UA_{h_j} \) is the number of user actions occurring by the end of hour \( j \) after the publication of the video. We generate a set of attention features for each user action, which we group into the following categories:

1. User actions daily percentage \((D_{d_i})\), or the fraction of user actions out of the total that occurred on day \( i \), where \( 1 \leq i \leq 7 \), and \( UA_{d_0} = 0 \) (7 features per user action):

\[
D_{d_i} = \frac{UA_{d_i} - UA_{d_{i-1}}}{UA_{d_7}}
\]

2. User actions daily cumulative percentage \((DC_{d_i})\), or the fraction of user actions out of the total by the end of day \( i \), where \( 1 \leq i \leq 7 \) (7 features per user action):

\[
DC_{d_i} = \frac{UA_{d_i}}{UA_{d_7}}
\]

\(^5\)http://developers.google.com/youtube/v3/docs/videos
3. User actions daily increase ($DI_d_i$), or the proportion of increase in the number of user actions on day $i$ compared to day $i - 1$, where $2 \leq i \leq 7$ (6 features per user action):

$$DI_d_i = \frac{UA_d_i - UA_{d_i-1}}{UA_{d_i-1}}$$

4. User actions hourly increase ($HI_{h_j}$), or the proportion of increase in the number of user actions during hour $j$ compared to those during hour $j - 1$, where $2 \leq j \leq 168$ (167 features per user action):

$$HI_{h_j} = \frac{UA_{h_j} - UA_{h_{j-1}}}{UA_{h_{j-1}}}$$

5. User actions average hourly increase per day ($AHI_{d_i}$), or the average hourly increase in the number of user actions on day $i$, where $1 \leq i \leq 7$ (7 features per user action):

$$AHI_{d_i} = \frac{\sum_{j=(i-1)*24+1}^{i*24} HI_{h_j}}{24}$$

6. User actions majority interval length ($MI_T$), or the number of hours containing the majority of the user actions

$$MI_T = \min_{1 \leq i, j \leq 168} \left\{ j - i \left| \frac{UA_{h_j} - UA_{h_i}}{UA_{h_{168}}} \geq T \right. \right\}$$

where $T$ (one of $\{0.5, 0.7, 0.9\}$) is the majority share (3 features per user action).

7. User actions peak delay interval ($PDI$), or the number of hours leading to the hour with the highest concentration of user actions (1 feature per user action):

$$PDI = \arg\max_{\{i|2 \leq i \leq 168\}} UA_{h_i} - UA_{h_{i-1}}$$

8. User actions alive interval length ($AI$), or the hour up to which user actions were recorded (1 feature per user action):

$$AI = \min_{1 \leq p \leq 167} (p|UA_{h_i} - UA_{h_{i-1}} = 0, \forall i \in \{p+1, \ldots, 168\})$$

9. User actions peak share ($PS$), or the number of user actions during the peak hour divided by the total (1 feature per user action):

$$PS = \max_{2 \leq i \leq 168} \left\{ \frac{UA_{h_i} - UA_{h_{i-1}}}{UA_{h_{168}}} \right\}$$

Figure 1: First day breakdown into periods

Most of the attention received by the videos in our corpus is concentrated in the first day after a video has been published, and thus we monitor the attention during this period more closely. Besides daily and hourly, we look at six additional periods during the first day, as depicted on Figure 1. We extract the following features: (i) Percentage of User Actions per Period, (ii) User Actions per Period Increase, and (iii) User Actions Average Hourly Increase for a Period. This yields 18 additional features and a total of 218 features per user action.

To model the opinion of the users regarding the videos, we also use features derived by ratios of different user action types:

- **Positive Reactions**: the ratio between the number of likes and the number of views;
- **Negative Reactions**: the ratio between the number of dislikes and the number of views;
- **Engagement**: the ratio between the number of comments and the number of views;
- **Controversiality**: the ratio between the number of likes and the sum of the number of likes and dislikes.

For each of these ratios, we calculate a set of features that show how the numbers change daily, similarly to the User Actions Daily Percentages (7 features per ratio) and the User Actions Daily Cumulative Percentages (6 features per ratio) features. We further generate ratio features for the more granular first day periods (6 features per ratio), which yields a total of 19 ratio-driven features. Overall, we have 952 attention features per video.

### 4.2 Models

Our architecture contains two consecutive classifications: (i) for YouTube videos, and (ii) for YouTube channels. As we want to make use of the features derived from the YouTube videos, we labelled each video with the factuality score of the channel that published it, using distant supervision.
Thus, the video classification learns to predict the factuality labels that are projected from the corresponding channels. Naturally, not all videos published by a low-factuality channel necessarily contain disinformation. Yet, this is not a problem since we do not aim at classifying correctly individual videos, but at detecting factuality-related patterns, which would then be used at the channel level: our channel classifier uses the predictions of the video-level classifier to predict the factuality of channels.

4.2.1 Video Modelling
For each video, we have 768 features from the sentence-level BERT representation. We calculated these features once for the title and once for the description of the video, obtaining a total of 1,536 textual features.

We further have 952 attention-driven features per video. To validate the relevance of these features with respect to our classification task, we apply a set of feature selection methods over the training split of our dataset, namely ANOVA, Pearson correlation, and Spearman correlation. According to these methods, the ratio features turn out to be the most relevant ones. We selected the best 100 features from each method or 124 attention video-level features, which we used in our classification experiments. Combined with the 1,536 textual features, this yielded a total of 1,660 features per video.

4.2.2 Channel Modelling
For the second classifier, at the channel level, we generate the following groups of features:

1. YouTube statistics (total of 13 features):
   - Popularity (7 features): number of subscribers, average number of hourly, daily, and weekly views and comments;
   - Activity (5 features): number of videos, average number of videos published hourly, daily, and weekly, and average number of videos per channel subscriber;
   - Attention concentration (1 feature): Gini index measuring the concentration of video views within a channel.

2. Averaged videos features (1,660) features): average values of the features for the videos published by the channel. For each video feature, we have a corresponding aggregated channel feature.

3. Aggregated video-level classifier predictions features (9 features): for each channel, we aggregate the predictions of the video-level classifier for the videos in that channel. We use three types of aggregations:
   - maximum probability across the videos for each factuality label;
   - average probability across the videos for each factuality label;
   - factuality distributions percents: for each factuality, this is the percent of videos predicted to have that factuality.

5 Experiments and Evaluation
We train two subsequent classifiers for factuality prediction: for videos and for channels. We conduct experiments with different models and we compare them to a majority-class baseline. We evaluate the models in terms of accuracy, balanced accuracy, and mean absolute error (MAE). MAE is a more relevant measure in our case as it takes into account the ordering of the labels: confusing high factuality with mixed factuality is a smaller error than confusing it with low factuality.

While our ultimate goal is to classify channels, we start with video classification, then we aggregate the predictions, and we use them to make predictions at the channel level.

We divide the dataset into training, development, and test split at the channel level. Then, for the video-level experiments, we use for training/development/testing the videos for the respective channels. Note that this guarantees that all videos for a given channel go into the same split.

5.1 Video-Level Classification
Below, we report results using Gradient Boosted Decision Trees (GBDT). We also experimented with logistic regression, ordinal logistic regression, and SVM with various kernels, but they performed worse. We trained separate models (a) using the textual representation, and (b) using the user attention cycles. Table 2 shows the evaluation results.

<table>
<thead>
<tr>
<th>#</th>
<th>Experiment</th>
<th>Acc.</th>
<th>Bal. Acc.</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Baseline</td>
<td>61.72</td>
<td>33.33</td>
<td>0.4391</td>
</tr>
<tr>
<td>1</td>
<td>BERT</td>
<td><strong>67.54</strong></td>
<td>45.89</td>
<td><strong>0.3692</strong></td>
</tr>
<tr>
<td>2</td>
<td>User attention</td>
<td>64.93</td>
<td><strong>55.12</strong></td>
<td>0.3979</td>
</tr>
</tbody>
</table>

Table 2: Video-level experiments with GBDT.
Table 3: Channel-level experiments.

<table>
<thead>
<tr>
<th>Group</th>
<th># Experiment</th>
<th>Dim.</th>
<th>Acc.</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>Majority class</td>
<td>-</td>
<td>63.08</td>
<td>0.4308</td>
</tr>
<tr>
<td><strong>Text</strong></td>
<td>1 BERT averaged</td>
<td>1,536</td>
<td>73.85</td>
<td>0.3077</td>
</tr>
<tr>
<td></td>
<td>2 BERT aggregated predictions</td>
<td>9</td>
<td>75.38</td>
<td>0.3077</td>
</tr>
<tr>
<td></td>
<td>3 BERT all</td>
<td>1,545</td>
<td>73.85</td>
<td>0.3538</td>
</tr>
<tr>
<td><strong>User Attention</strong></td>
<td>4 User attention averaged</td>
<td>124</td>
<td>75.38</td>
<td>0.2769</td>
</tr>
<tr>
<td></td>
<td>5 User attention channel statistics</td>
<td>13</td>
<td>63.08</td>
<td>0.4000</td>
</tr>
<tr>
<td></td>
<td>6 User attention aggregated predictions</td>
<td>9</td>
<td>67.69</td>
<td>0.3846</td>
</tr>
<tr>
<td></td>
<td>7 User attention all</td>
<td>146</td>
<td>70.77</td>
<td>0.3231</td>
</tr>
<tr>
<td><strong>Ensemble</strong></td>
<td>8 BERT all + User attention averaged</td>
<td>1,669</td>
<td>76.92</td>
<td>0.2615</td>
</tr>
</tbody>
</table>

Note that our datasets are not well-balanced and have very few examples of low-factuality videos and channels. To mitigate this, we apply oversampling using SMOTE (Chawla et al., 2002), which generates additional synthetic examples. Moreover, it is important that the video classifier generates predictions for the low-factuality and the mixed-factuality classes; otherwise, the predictions for these classes could be lost when aggregating for the channel classification. Thus, we also report balanced accuracy, as it is important when choosing which video experiments to select for aggregation.

5.2 Channel-Level Classification

We experimented with several approaches for channel classification:
- using aggregated video-level features to obtain channel-level representation;
- using the posterior probabilities of video-level classifiers;
- using the previous two together;
- ensemble of different channel-level classifiers.

For the ensemble aggregation, we experimented with three methods for choosing the most likely class for a given channel:
- after averaging the predictions from the various models (mean);
- after getting the maximum probability prediction from the various models (max);
- after getting the minimum probability prediction from the various models (min) – tells us which class is least likely to be wrong.

The results are shown in Table 3. All experiments use GBDT, except for experiment 8, which uses ordinal logistic regression.

6 Discussion

Below, we analyze the results and we perform an ablation study.

6.1 Result Analysis

We can see in Table 3 that all models improve over the majority class baseline by a sizable margin. We further see that using average information on user attention cycles (line 4) performs better than using textual features (lines 1–3). Moreover, combining the two yields the best result (line 8). The other two sets of user attention features: channel statistics and aggregated video-classifier predictions, do not contribute to the combined user attentions model (compare line 4 to line 6).

The relative improvements over the majority class baseline in terms of accuracy are generally smaller than those for MAE, which can be explained by class imbalance. To improve accuracy, the models need to learn to assign mixed and low factuality labels properly (as the majority class is high factuality). Most of the trained models undervalue the unrepresented classes. If some of the balancing techniques are applied, the models recognize better the low and the mixed examples, but at the cost of false positives for these classes from the high-factuality examples, which decreases the overall accuracy. In contrast, MAE rewards models that can improve the small classes even given the risk of introducing some errors for the majority class.
Table 4: Ablation study (channel-level classification) using various user attention features.

<table>
<thead>
<tr>
<th>Feature Group</th>
<th>Acc.</th>
<th>Bal. Acc.</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>63.08</td>
<td>33.33</td>
<td>0.4308</td>
</tr>
<tr>
<td>Views (V)</td>
<td>66.15</td>
<td>41.79</td>
<td>0.3692</td>
</tr>
<tr>
<td>Dislikes (D)</td>
<td>64.62</td>
<td>39.27</td>
<td>0.4000</td>
</tr>
<tr>
<td>Comments (C)</td>
<td>56.92</td>
<td>32.64</td>
<td>0.4615</td>
</tr>
<tr>
<td>Likes (L)</td>
<td>64.62</td>
<td>37.56</td>
<td>0.3846</td>
</tr>
<tr>
<td>V + L + C</td>
<td>67.69</td>
<td>49.27</td>
<td>0.3385</td>
</tr>
<tr>
<td>V + L + D + C</td>
<td>69.23</td>
<td>44.27</td>
<td>0.3385</td>
</tr>
<tr>
<td>Engagement</td>
<td>63.08</td>
<td>44.27</td>
<td>0.4000</td>
</tr>
<tr>
<td>Controversiality</td>
<td><strong>70.77</strong></td>
<td>48.33</td>
<td><strong>0.3231</strong></td>
</tr>
<tr>
<td>Positive reactions</td>
<td><strong>70.77</strong></td>
<td>48.33</td>
<td><strong>0.3231</strong></td>
</tr>
<tr>
<td>Contr + Eng</td>
<td>63.08</td>
<td>45.12</td>
<td>0.4000</td>
</tr>
<tr>
<td>Contr + Pos</td>
<td>67.69</td>
<td>46.71</td>
<td>0.3538</td>
</tr>
<tr>
<td>Pos + Eng</td>
<td>64.62</td>
<td>46.79</td>
<td>0.3846</td>
</tr>
<tr>
<td>Channel statistics</td>
<td>63.08</td>
<td>39.31</td>
<td>0.400</td>
</tr>
<tr>
<td>Aggregated</td>
<td>67.69</td>
<td>60.20</td>
<td>0.3846</td>
</tr>
</tbody>
</table>

6.2 Ablation Study

As our focus is on attention cycles, we performed an ablation study for these features against the combined user attention channel model. The results are shown in Table 4. We can see that ratio features such as controversiality and positive reactions alone yield the best accuracy and MAE. Using the predictions of the video-level classifier as features yields the best balanced accuracy. This confirms the importance of having accurate low- and mixed-factuality predictions for the video classifier prior to the aggregation. Finally, the overall best results, when considering all measures, are achieved when combining all features.

7 Conclusion and Future Work

We proposed a novel framework for predicting the factuality of reporting of news outlets by studying the user attention cycles in their respective YouTube channels. We further designed a rich set of features derived from the temporal evolution of the number of views, likes, dislikes, and comments for a video, which we then aggregated at the channel level. Our experiments demonstrated both complementarity and sizable improvements over state-of-the-art textual representations.

We further developed and released a dataset containing observations about user attention on YouTube channels for 489 news media. We hope that this will enable future research on using data from video sharing platforms.

In future work, we plan to improve the class imbalance of the dataset by extending it with more examples. We further want to integrate additional features based on the comments for the videos and other information sources such as Twitter and Wikipedia. Finally, we plan to study the utility of user attention cycles for other related tasks such as political ideology detection for news media.

Ethics and Broader Impact

Data Collection Our dataset was collected from YouTube, using their public API.

User Privacy Our dataset contains aggregated attention statistics without any user data.

Biases Any biases found in the dataset are unintentional, and we do not intend to do harm to any group or individual.

Intended Use and Misuse Potential Our dataset and the proposed model can enable the development of systems for automatic detection of reliable/unreliable YouTube channels, which could support media literacy, as well as analysis and decision making for the public good. However, they could also be misused by malicious actors.

Environmental Impact Finally, we would also like to warn that the use of large-scale Transformers requires a lot of computations and the use of GPUs/TPUs for training, which contributes to global warming (Strubell et al., 2019).

Acknowledgments

This research is part of the Tanbih mega-project, developed at the Qatar Computing Research Institute, HBKU, which aims to limit the impact of “fake news”, propaganda, and media bias by making users aware of what they are reading, thus promoting media literacy and critical thinking.

This research is also partially supported by Project UNITe BG05M2OP001-1.001-0004 funded by the OP “Science and Education for Smart Growth” and co-funded by the EU through the ESI Funds.

6http://tanbih.qcri.org
References


Benjamin D. Horne and Sibel Adali. 2017. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. ArXiv 1703.09398.


Deep CNN–LSTM hybrid neural networks have proven to improve the accuracy of Optical Character Recognition (OCR) models for different languages. In this paper we examine to what extent these networks improve the OCR accuracy rates on Swedish historical newspapers. By experimenting with the open source OCR engine Calamari, we are able to show that mixed deep CNN–LSTM hybrid models outperform previous models on the task of character recognition of Swedish historical newspapers spanning 1818–1848. We achieved an average character accuracy rate (CAR) of 97.43% which is a new state–of–the–art result on 19th century Swedish newspaper text. Our data, code and models are released under CC BY licence.

2 Related Work

Mainly due to the introduction of recurrent neural networks (RNNs) in particular with the Long Short–Term Memory (LSTM) architecture (Breuel, 2017) great progress has been made in the field of OCR in recent years. Today most of modern OCR systems leverage deep learning algorithms for training models and to improve performance. For instance some state-of-the-art OCR systems use shallow LSTM neural networks, consisting of one or two hidden layers (Breuel, 2008). LSTM models became a standard in 2013 (Breuel et al., 2013) and LSTM-based networks have proven to be one of the most effective approaches for complex natural language processing (NLP) tasks (Hochreiter and Schmidhuber, 1997; Alom et al., 2019). Likewise Convolutional Neural Networks (CNNs) have shown outstanding results for image data processing tasks (Krizhevsky et al., 2017) including

1https://pdf.abbyy.com

2Data, code and models are released under CC BY licence:
https://github.com/mskelb/EXJOBB
feature extraction (Lecun and Bengio, 1995). Hybrids of these two network structures are used in a high diversity of fields and have achieved state-of-the-art results in many cases (Wick et al., 2018).

Breuel (2017) presented a groundbreaking deep hybrid CNN–LSTM implementation for text recognition, that outperformed previously state of the art methods based on shallow LSTMs. Reul et al. (2017) show that transfer learning drastically improves character accuracy rates on early printed books, compared to when training models from scratch. A year later this have become the default approach for training models. In their continued research Reul et al. (2018) utilize a combination of cross-fold training and confidence voting and succeed to significantly reduce character error rates on early printed books, compared to training a single model on a single fold. Along the same lines, in our work we also train an ensemble of models on a cross-fold of the same GT data and then combine the models through voting.

Drobac and Lindén (2020) and Wick et al. (2018) approach the complex task of OCR for historical prints by utilizing these types of deep CNN–LSTM hybrid networks. To improve OCR results for historical Finnish and Swedish newspapers and journals Drobac and Lindén (2020) train mixed-language models and after post-processing of the OCR output achieve 1.7% CER for the Finnish, and 2.7% CER for the Swedish test set from 1771 to 1874. Unlike Drobac and Lindén (2020) we do not apply any post-processing method but rather focusing on increasing the accuracy of the Swedish character model.

In Wick et al. (2018) error–rates are successfully reduced to a factor of up to 55% for digitised historical texts from the ICDAR 2017 dataset, achieving an average CER of 1.5%. To further improve these results, confidence voting was applied, resulting in CER below 0.5%. Moreover, Wick et al. (2018) show these types of deep neural networks significantly outperform shallow networks in terms of both recognition capabilities and speed.

3 Calamari Deep CNN–LSTM Hybrid Networks

Calamari is a high–performance Tensorflow–based package for line based recognition using state of the art Deep Neural Networks (DNNs) (Wick et al., 2020). The advantage of Calamari is that the software supports customized deep network architectures composed by Convolutional Neural Networks (CNNs) and Long Short–Term Memory (LSTM) layers, trained by the Connectionist Temporal Classification (CTC) algorithm described in Graves et al. (2006). It uses the Tensorflow 3 framework for deep neural network computations, and consequently supports training and recognition on the graphics processing unit (GPU), which is proven to significantly reduce the overall computation time (Wick et al., 2020).

Calamari also provides additional features that might improve accuracy rates, such as pre–training, early stopping, cross–fold training, data argumentation, and confidence voting of different predictions. However, while data augmentation has been shown to improve accuracy for small data sets, it impairs the accuracy for larger data sets according to Wick et al. (2020). Moreover, Calamari with its deep network architecture has proven to significantly outperforms the shallow one–dimensional LSTM based neural network approach, such as the one used by OCRopus (Breuel, 2008) in terms of both recognition capabilities and speed (Drobac and Lindén, 2020; Wick et al., 2018).

The default neural network structure consists of two consecutive CNN blocks containing a convolution and a max–pooling layer, both connected with a ReLU–activation function. Each of the two convolution layers has a kernel size of 3 x 3, the first one consists of 40 filters, and the second one of 60 filters. The pooling layer has a kernel size and stride of 2 x 2. This is followed by a bidirectional LSTM layer, and finally an output layer with a drop–out rate of 0.5 in order to prevent overfitting. Given the predictions of the output–layer and the GT labels, the CTC–loss is computed.

4 Data and Preprocessing

The reference data we experimented with forms part of the KubHist data (Adesam et al., 2019). It is a large collection of approximately 300 thousand Swedish historical newspaper editions, spread over roughly 200 years. The collection has been digitised and OCRed by the National Library of Sweden.

Part of the KubHist, more specifically a reference data of 400 pages spanning between 1818 and 2018, was processed through an enhanced OCR–process (Dannélls et al., 2019) by combin-
ing two OCR engines: Abbyy FineReader, and the open source system Tesseract. This two–OCR engine system was originally developed in cooperation with the Norwegian software company Zissor in 2017 and is based on the principle of evaluating and comparing the OCR results from multiple engines. The approach has been proven to improve the character recognition accuracy for some newspapers (Dannélls et al., 2020). Many errors however remain. The reference data comes with a ground truth (GT) data available in plain text files. It was first segmented down to paragraph level, and further manually transcribed through double–keying.

This reference material has been previously used to evaluate the two–OCR engine system, focusing on the time period 1818–2018. The evaluation results show that accuracy varies from 56.65% to 98.41%, on character level, depending on the newspaper edition. These results further constitute the very baseline of this project.

4.1 Challenges

The challenge of the KubHist corpus lies in the amount of OCR errors that are currently very high in some parts of the collection. Especially texts from the early 19th century where it is estimated that most of the text (roughly 75%) is printed in Blackletter. The Blackletter typeface is generally challenging for OCR systems to recognize due to the many font variations, low distinctiveness of characters, and in many cases, lack of training data of acceptable quality (Holley, 2009; Furrer and Volk, 2011). Large part of the OCR errors is a result of the low accuracy of the pre–trained language models that were trained on a limited amount of Swedish data and were used in the OCR process.

4.2 Data Preprocessing

To test the performance of our mixed deep CNN–LSTM hybrid models, part of the avail-

able reference material has been used for training, validation and testing respectively. This dataset we experimented with contains two pages from each newspaper edition during the time period 1818–1848. In all 30 newspaper editions, one from each year, and 60 newspaper pages that have been further pre–processed through image binarization and de–skewing using the ocropus-nbin script provided by OCRopus. For the binarization procedure adaptive thresholding has been applied. All newspaper pages have been further re–segmented into text line images using the ocropus-gpageseg script, also provided by OCRopus, comprising a total of 8 413 lines, 67 441 words and 423 414 characters. To maintain the data as clean as possible incorrectly segmented text lines as well as non textual content such as vertical and horizontal lines etc. have been manually removed.

5 Experiments and Results

Our strategy for dividing the dataset into training and test sets has been to randomly selecting lines from each newspaper page. 80% has been allocated for training and 20% for testing. This way of randomizing generates a good diverse training data and hence a good representation of the data that is important for constructing a model able to generalize well (Drobac and Lindén, 2020). In total, 6 742 lines containing 53 963 words and 338 701 characters have served as training set, 1 671 lines containing 13 478 words and 84 713 characters as test set. For training of a single model (single voter) the final training set has been further split into training and validation subsets, again with a split ratio of 80:20. In total the generated training subset comprises 5 400 lines and the validation set 1 349 lines. More details can be found in Table 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>GT lines</th>
<th>Words</th>
<th>Characters</th>
<th>B:A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>6 742</td>
<td>53 963</td>
<td>338 701</td>
<td>75% : 25%</td>
</tr>
<tr>
<td>training set</td>
<td>5 400</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>validation set</td>
<td>1 349</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>1 671</td>
<td>13 478</td>
<td>84 713</td>
<td>75% : 25%</td>
</tr>
<tr>
<td>Total</td>
<td>8 413</td>
<td>67 441</td>
<td>423 414</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Details of the training, validation and test sets. B:A is the distribution of Blackletter and Antiqua.
Once the line images have been prepared we use them together with their ground truth to train models with Calamari. During training early stopping has been applied in order to avoid overfitting and improve generalization. If performance on the held out validation dataset after each epoch has not improved after 5 epochs or if validation loss has begun increasing training was stopped. One approach for reducing overfitting is to add more examples to the training data. However, since segmenting newspaper pages into image lines and linking them to their corresponding GT lines has turned out to be relatively time consuming. Adding more data in order to investigate whether results can be improved has not been done.

To improve performance and find the optimal neural network structure for our data a series of experiments have been performed using different network configurations. These include testing out different combinations and dimensions of the CNN and LSTM layers and in this way expanding the network. These experiments are described in the following sections. In all our experiments both single models are trained as well as using a combination of cross-fold training and subsequent confidence voting. Training was performed using a 5–fold over the same training data resulting in 5 models with different characteristics. These 5 models are then used to recognize the held out test lines. For each test line a total of 5 output sequences are generated, 1 from each model. These 5 output sequences later serve as the input for the voting process in order to determine the final output by using confidence scores.

After experimenting with different network configurations both by training single models and by combining cross-fold training and confidence voting we found that the best results were achieved using the following neural network, seen in Equation 1.

\[
\begin{align*}
\text{cnn} &= 80 : 3 \times 3, \text{pool} = 2 \times 2, \\
\text{cnn} &= 100: 3 \times 3, \text{pool} = 2 \times 2, \\
\text{lstm} &= 200, \text{dropout} = 0.5, \\
\text{lstm} &= 200, \text{dropout} = 0.5.
\end{align*}
\]

(1)

The optimized neural network consists of two consecutive CNN blocks containing a convolution layer followed by a max–pooling layer. Both connected with a ReLU–activation function. Each of the two convolution layers have a kernel size of 3 x 3. The first one consists of 80 filters and the second one of 100 filters. The pooling layer has a kernel size and stride of 2 x 2. The two CNN blocks are followed by two LSTM layers containing a total of 400 nodes divided over two layers with 200 nodes each.

5.1 Experimental Setup

We ran four experiments using the following model combinations:

- **M1**: single model trained using Drobac and Lindén (2020) neural network in Equation 3.
- **M2**: single model trained using Calamari’s default neural network in Equation 2.
- **M3**: single model trained using the optimized neural network in Equation 1.
- **M4**: 5 best models trained using cross-fold training with the same optimized neural network in combination with subsequent confidence voting.

**Experiment I** In this initial experiment we train a single model using Calamari’s default neural network as specified in Equation 2

\[
\begin{align*}
\text{cnn} &= 40 : 3 \times 3, \text{pool} = 2 \times 2, \\
\text{cnn} &= 60: 3 \times 3, \text{pool} = 2 \times 2, \\
\text{lstm} &= 200, \text{dropout} = 0.5.
\end{align*}
\]

(2)

**Experiment II** We further train a single model following the same network architecture as in Drobac and Lindén (2020) and in Equation 3. Similar to the default Calamari network this network contains two consecutive CNN blocks containing a convolution and a max–pooling layer. Both are connected with a ReLU–activation function as can be seen in Equation 4.2. Each of the two convolution layers has a kernel size of 3 x 3. The first one consists of 128 filters and the second one of 128 filters. The pooling layer has a kernel size and stride of 2 x 2. In addition, the LSTM layers contain a total of 1 200 nodes divided over two layers with 600 nodes each. Here we want to investigate how performance changes using a deeper network.

\[
\begin{align*}
\text{cnn} &= 128 : 3 \times 3, \text{pool} = 2 \times 2, \\
\text{cnn} &= 128: 3 \times 3, \text{pool} = 2 \times 2, \\
\text{lstm} &= 600, \text{dropout} = 0.5. \\
\text{lstm} &= 600, \text{dropout} = 0.5.
\end{align*}
\]

(3)

**Experiment III** In this experiment we want to find the optimal neural network for our data. In
Table 2: CAR results for the four different models. Respective model combinations: M1–Drobac, M2–Calamari, M3–Single model trained with the optimized network, M4–5 best models trained with the optimized network.

<table>
<thead>
<tr>
<th>Model(s)</th>
<th>Dataset</th>
<th>CAR (%)</th>
<th>CER (%)</th>
<th>Voted</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) M1</td>
<td>Test set</td>
<td>95.55</td>
<td>4.45</td>
<td>No</td>
</tr>
<tr>
<td>(2) M2</td>
<td>Test set</td>
<td>96.06</td>
<td>3.94</td>
<td>No</td>
</tr>
<tr>
<td>(3) M3</td>
<td>Test set</td>
<td>96.44</td>
<td>3.56</td>
<td>No</td>
</tr>
<tr>
<td>(4) M4</td>
<td>Test set</td>
<td><strong>97.43</strong></td>
<td><strong>2.57</strong></td>
<td>Yes</td>
</tr>
<tr>
<td>(5) M4</td>
<td>50-fraktur</td>
<td>96.84</td>
<td>3.16</td>
<td>Yes</td>
</tr>
<tr>
<td>(6) M4</td>
<td>50-antiqua</td>
<td>96.48</td>
<td>3.52</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Reul et al. (2018) it was shown that a combination of cross-fold training and confidence voting led to significantly lower character error rates compared to training a single model. Therefore to test the models’ ability of predicting never seen before data we have followed the same approach in our continued experiments. However, in order to investigate how much this combination can improve performance models have been trained both by (1) single training resulting in a single voter and (2) by cross-fold training resulting in 5 models.

Experiment IV This experiment investigates the transferability of models and their ability to generalize on previously unseen data. In this case unseen data is a collection consisting of journals and newspaper published in Finland between 1771 and 1874. Two relatively small datasets of text line images and their corresponding GT have been randomly sampled from the Swedish test dataset in Drobac and Lindén (2020). The first set contains 50 lines printed in Fraktur and the second 50 lines printed in Antiqua. Models have been tested on each separate test set using confidence voting. What has been interesting to see here is how mixed models behave on a specific font type.

5.2 Evaluation Metrics

Calamari uses the Character Error Rate (CER) metric defined as the “edit distance (ed) of two sequences $s_1$ and $s_2$ normalized by the maximum length” (Wick et al., 2020) as stated in Equation 4. Edit distance corresponds to the Levenshtein distance and the sequences to the text lines and the corresponding GT lines. There are two common ways of measuring OCR errors or accuracy. One of which is at the character level (CER or CAR) and the other one at the word level (WER or WAR) (Holley, 2009).

$$CER = \frac{ed(s_1, s_2)}{\max(\|s_1\|, \|s_2\|)}$$ (4)

While CER is used for measuring performance of our models CAR has previously been used in the evaluation process of the two–OCR engines in Dannélls et al. (2020). In order to give a fair comparison of results a conversion has been made between CER and CAR. Moreover, when testing an ensemble of 5 models on the held out test lines we use the confidence based voting to combine them.

5.3 Results

Table 2 summarises the performance in average CAR (%) and CER (%) of the different combinations of the trained models tested on different datasets. Models in (1)–(4) have been evaluated using our test data. Models in (5) have been evaluated using 50 lines of Fraktur and in (6) 50 lines of Antiqua has been used originating from the Swedish test set in Drobac and Lindén (2020).

As Table 2 shows the test results trained with a single model (M2) reveals a significant improvement in CAR. An average 96.06% CAR was achieved corresponding to 3.94% CER. This can be compared to our baseline with an average CAR of 82.37% in Dannélls et al. (2020) for the specific time period.

The test results of M1 shows slightly lower CAR of 95.55% corresponding to 4.45% CER compared to using Calamari’s default network.

The best CAR was achieved when voting with 5 models (M4 in Table 2) originating from 5-fold training using the neural network in Equation 1. We achieved CAR of 97.43% corresponding to CER of 2.57% after evaluation on the test set.

Training a single model (M3 in Table 2) with the same neural network and then evaluating on the test set resulted in a slightly lower CAR of 96.44%
Table 3: CAR for each newspaper edition spanning between 1818–1848. The second column show previously obtained results from the evaluation of the two OCR–engine system in Dannéls et al. (2020), which constitutes the baseline for this project. Third column show results obtained after cross-fold training and voting using the neural network architecture in Equation 1. The forth column show the different in percentage units between the baseline and our results.

<table>
<thead>
<tr>
<th>NEWSPAPER EDITION</th>
<th>BASELINE</th>
<th>BEST</th>
<th>+/- (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>STOCKHOLMSPOSTEN 1818-09-23</td>
<td>61.08</td>
<td>96.87</td>
<td>+35.79</td>
</tr>
<tr>
<td>GÖTHEBORGS ALLEHANDA 1819-05-26</td>
<td>56.65</td>
<td>94.79</td>
<td>+38.14</td>
</tr>
<tr>
<td>CARLSCRONAS WEKOBLAD 1820-05-10</td>
<td>78.59</td>
<td>94.89</td>
<td>+16.30</td>
</tr>
<tr>
<td>GÖTHEBORGSKA NYHETER 1821-10-27</td>
<td>80.84</td>
<td>94.95</td>
<td>+14.11</td>
</tr>
<tr>
<td>GÖTHEBORGS TIDNINGAR 1822-12-10</td>
<td>70.28</td>
<td>96.46</td>
<td>+26.18</td>
</tr>
<tr>
<td>DAGLIGT ALLEHANDA 1823-02-20</td>
<td>83.56</td>
<td>98.35</td>
<td>+14.79</td>
</tr>
<tr>
<td>WEXJÖBLADET 1824-08-21</td>
<td>80.55</td>
<td>98.85</td>
<td>+18.30</td>
</tr>
<tr>
<td>POST- OCH INRIKES TID. 1825-06-16</td>
<td>81.03</td>
<td>96.18</td>
<td>+15.15</td>
</tr>
<tr>
<td>STOCKHOLMS DAGBLAD 1826-11-27</td>
<td>78.76</td>
<td>96.03</td>
<td>+17.27</td>
</tr>
<tr>
<td>AFTONBLADET 1827-07-02</td>
<td>68.09</td>
<td>93.20</td>
<td>+25.11</td>
</tr>
<tr>
<td>HELSBINGBORGSPOSTEN 1828-01-29</td>
<td>84.80</td>
<td>98.65</td>
<td>+13.85</td>
</tr>
<tr>
<td>DAGLIGT ALLEHANDA 1829-04-28</td>
<td>68.69</td>
<td>98.44</td>
<td>+29.75</td>
</tr>
<tr>
<td>NORRKÖPINGS TIDNINGAR 1830-03-30</td>
<td>81.97</td>
<td>98.44</td>
<td>+16.47</td>
</tr>
<tr>
<td>POST- OCH INRIKES TID. 1831-12-16</td>
<td>77.21</td>
<td>97.05</td>
<td>+19.84</td>
</tr>
<tr>
<td>GÖTEBORGS HAND. OCH SJÖ. 1832-10-15</td>
<td>92.15</td>
<td>97.92</td>
<td>+5.77</td>
</tr>
<tr>
<td>GÖTHEBORGS ALLEHANDA 1833-08-30</td>
<td>84.21</td>
<td>98.97</td>
<td>+14.76</td>
</tr>
<tr>
<td>MALMÖ ALLEHANDA 1834-03-12</td>
<td>87.41</td>
<td>98.34</td>
<td>+10.93</td>
</tr>
<tr>
<td>WEXJÖBLADET 1835-09-18</td>
<td>87.88</td>
<td>98.34</td>
<td>+10.46</td>
</tr>
<tr>
<td>POST- OCH INRIKES TID. 1836-12-08</td>
<td>96.40</td>
<td>98.46</td>
<td>+2.06</td>
</tr>
<tr>
<td>GEFLERBORGST LÄNS TIDNING 1837-02-01</td>
<td>82.15</td>
<td>98.49</td>
<td>+16.34</td>
</tr>
<tr>
<td>FREJA 1838-05-18</td>
<td>95.48</td>
<td>98.08</td>
<td>+3.32</td>
</tr>
<tr>
<td>CARLSCRONAS WEKOBLAD 1839-09-25</td>
<td>74.54</td>
<td>97.74</td>
<td>+23.2</td>
</tr>
<tr>
<td>SVENSKA BIET 1840-12-16</td>
<td>97.08</td>
<td>97.24</td>
<td>+0.16</td>
</tr>
<tr>
<td>NAJADEN 1841-09-03</td>
<td>95.36</td>
<td>95.78</td>
<td>+0.58</td>
</tr>
<tr>
<td>NORRLANDSPOSTEN 1842-01-21</td>
<td>85.97</td>
<td>96.95</td>
<td>+10.96</td>
</tr>
<tr>
<td>GÖTEBORGS HAND. OCH SJÖ. 1843-06-03</td>
<td>91.10</td>
<td>98.02</td>
<td>+6.93</td>
</tr>
<tr>
<td>WERMLANDSTIDNINGEN 1844-06-05</td>
<td>85.08</td>
<td>98.04</td>
<td>+12.96</td>
</tr>
<tr>
<td>JÖNKÖPINGSBLADET 1845-05-03</td>
<td>86.0</td>
<td>97.74</td>
<td>+11.74</td>
</tr>
<tr>
<td>NERIKES ALLEHANDA 1846-03-18</td>
<td>85.29</td>
<td>96.76</td>
<td>+11.44</td>
</tr>
<tr>
<td>GÖTHEBORGSKA NYHETER 1847-07-24</td>
<td>86.00</td>
<td>98.65</td>
<td>+12.65</td>
</tr>
<tr>
<td>SNÄLLPOSTEN 1848-05-22</td>
<td>89.41</td>
<td>99.73</td>
<td>+10.32</td>
</tr>
</tbody>
</table>

Average 82.37% 97.43% +15.06
corresponding to a CER of 3.56%. Compared to using voting this is nearly a 1% drop in CAR corresponding to approximately 827 more errors in the output. In total the best CAR of 97.43% is an improvement of 15.06% in comparison to the baseline. The results for each individual newspaper edition can be found in Table 3. In total CAR has improved for each individual newspaper edition.

All results were achieved through voting by the 5 models trained using the optimized neural network in Equation 1. Evaluation of the results on the test dataset containing 50 lines of Fraktur showed 96.84% CAR corresponding to 3.16% CER, displayed at row (5) in Table 2. On the test set containing Antiqua a slightly lower CAR of 96.44% was achieved corresponding to 3.56% CER.

5.4 Error Analysis

Table 4 show confusion matrices over the 10 most common character errors made by the single model (left) and after voting using an ensemble of 5 models (right). All models were trained using the optimized neural network in Equation 1. The most common confusions are in both cases as expected similar characters, such as “¨a” and “a”, “¨o” and “o” and the confusion between long–s “ſ” and “f”. Most notable when using voting the number of confusions of “¨o” and “o” dropped from 95 to 52 occurrences and the deletion of a space dropped from 53 to below 10 occurrences. In summary, all confusions were reduced using voting. However, the confusions between “¨a” and “a”, and “¨o” and “o” still constitute the most frequent confusions.

6 Conclusion and Future Work

In this paper we have investigated how deep CNN–LSTM hybrid neural networks can be utilized in order to improve current OCR results for 19th century Swedish newspaper text. Mixed deep CNN–LSTM hybrid models have been successfully trained in Calamari for the task of character recognition of Swedish historical newspaper texts spanning 1818–1848. Initial testing with the Calamari default network revealed a significant improvement in accuracy compared to our baseline (avg. 82.37% CAR) resulting in a CAR of 96.06%. This test alone showed the advantage of training individual mixed models over pre–trained models from commercial systems such as ABBYY FineReader. Highest CAR of 97.43% was achieved through voting with 5 best models using the optimized network. Thus, the combination of cross-fold training and confidence based voting significantly improve accuracy rates compared to training a single model using the same neural network. Furthermore, our best results show a significant improvement over the baseline results for the specific time period.

A promising future direction is to incorporate active learning (Reul et al., 2018). Active learning is based on the principle of maximal disagreement. An ensemble of voters (or models) are first trained on a set of training lines with their corresponding GT. Then, the voters are given unseen text lines and make predictions. Those lines where voters disagree the most on is then added to the training data for subsequent training enabling a maximal learning effect.
Acknowledgments

This work has been funded by the Swedish Research Council as part of the project Evaluation and refinement of an enhanced OCR-process for mass digitisation (2019–2020; dnr IN18-0940:1).

It is also supported by Språkbanken Text and Swe-Clarin, a Swedish consortium in Common Language Resources and Technology Infrastructure (CLARIN) Swedish CLARIN (dnr 821-2013-2003). The authors would like to thank the RANLP anonymous reviewers for their valuable comments.

References


A Psychologically Informed Part-of-Speech Analysis of Depression in Social Media

Ana-Maria Bucur1 Ioana R. Podina2 Liviu P. Dinu3
1 Interdisciplinary School of Doctoral Studies
2 Laboratory of Cognitive Clinical Sciences, Department of Psychology
3 Faculty of Mathematics and Computer Science
University of Bucharest, Romania
ana-maria.bucur@drd.unibuc.ro, ioana.podina@fpse.unibuc.ro
ldinu@fmi.unibuc.ro

Abstract

In this work, we provide an extensive part-of-speech analysis of the discourse of social media users with depression. Research in psychology revealed that depressed users tend to be self-focused, more preoccupied with themselves and ruminate more about their lives and emotions. Our work aims to make use of large-scale datasets and computational methods for a quantitative exploration of discourse. We use the publicly available depression dataset from the Early Risk Prediction on the Internet Workshop (eRisk) 2018 and extract part-of-speech features and several indices based on them. Our results reveal statistically significant differences between the depressed and non-depressed individuals confirming findings from the existing psychology literature. Our work provides insights regarding the way in which depressed individuals are expressing themselves on social media platforms, allowing for better-informed computational models to help monitor and prevent mental illnesses.

1 Introduction

Mental health disorders are a common problem in our world. Currently, mental health issues are on the rise: there is a 13% increase in the past decade according to World Health Organization (WHO)1, with depression being at the forefront. Many mental illnesses remain undiagnosed due to social stigma, leading people to live 1 in 5 years of disability in their lifetime.

With the rise of social media websites, interdisciplinary researchers in natural language processing, psychology and network analysis have turned their attention to automatically detect and monitor mental health manifestations through users’ individual activity on social media platforms (e.g. Facebook, Twitter, Reddit). The research is primarily focused on analyzing users’ texts from posts and comments and determining, through computational linguistics models, the risk for various mental conditions - self-harm, depression, addictions etc.

Research is fulfilled through curated datasets (Yates et al., 2017; Losada and Crestani, 2016; Amir et al., 2019) with texts from primarily Reddit and Twitter. At the forefront of incentivising interdisciplinary research on monitoring mental health on social media are workshops such as the Early risk prediction on the Internet (eRisk) Workshop2 and the Workshop on Computational Linguistics and Clinical Psychology (CLPsych)3.

CLPsych and eRisk are two significant initiatives focusing on the interdisciplinary research area between computational linguistics and psychology. The eRisk Workshop, from the Conference and Labs of the Evaluation Forum (CLEF), focuses on the technologies that can be used for early risk detection of different pathologies or safety threats (Losada et al.). In five years of existence, the workshop addressed multiple mental health problems: pathological gambling, depression, self-harm and anorexia.

The CLPsych Workshop was co-located with several international conferences on natural languages processing, the last edition was co-located with the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL). Throughout the seven editions of this workshop, it hosted shared tasks on depression, post-traumatic stress disorder (PTSD) (Coppersmith et al., 2015b), labeling crisis posts from the peer-support forum ReachOut4 (Milne et al., 2016), predicting current and future psychological health from childhood essays (Lynn et al., 2018), the degree of suicide risk (no risk, low, moderate, no risk, low, moderate,
or severe risk) (Zirikly et al., 2019) and suicide risk prediction from real data donated through OurDataHelps5 (Goharian et al., 2021).

In the present study, we perform a part-of-speech analysis and contribute to the understanding of the differences in social media discourse between depressed and non-depressed individuals. We focus on the differences in part-of-speech use and ground them in the existing literature from psychology researchers. We aim to answer the following two research questions:

RQ1: Are there significant differences in part-of-speech use between individuals with self-reported depression diagnosis and control?

RQ2: Can these part-of-speech features be used alone to differentiate individuals with depression from controls?

2 Related Work

Detecting the manifestations of mental health disorders from social media is an interdisciplinary research problem for researchers from psychology, natural language processing and network analysis. The two main approaches used to detect cues of depression from online users are: extracting linguistic features for a quantitative analysis or using automated models for classification.

The differences in language between depressed and non-depressed individuals focus on greater use of negative words, the personal pronoun "I" (Rude et al., 2004), more words with negative polarity, frequent dichotomous expressions (e.g. always, never) (Fekete, 2002), cues of rumination (reflected in greater use of past tense verbs) (Smirnova et al., 2018) in texts from depressed individuals.

Most linguistic features are extracted using the Linguistic inquiry and word count (LIWC) lexicon (Pennebaker et al., 2001). LIWC provide a list of dictionary words for more than seventy categories: part-of-speech (e.g. personal pronouns, first-person personal pronouns, nouns, present/future/past verbs, adjectives), psychological processes (e.g. social, affective, cognitive processes), personal concerns (e.g. work, money, religion, death), etc. It is used to detect cues of depression (Loveys et al., 2018; Nalabandian and Ireland, 2019; Eichstaedt et al., 2018), neuroticism (Resnik et al., 2013), to explore the language of suicide poets (Stirman and Pennebaker, 2001), etc.

Other approaches to mental illnesses detection from text rely on character and word n-grams (Coppersmith et al., 2015a; Pedersen, 2015) or topic modelling techniques (Preotiuc-Pietro et al., 2015; Bucur and Dinu, 2020).

Recent studies analyzing the online discourse of social media users with depression have focused on other particularities of language, such as offensive language. Birnbaum et al. (2020) found that depressed individuals use more swear words in their Facebook messages compared to controls. Bucur et al. (2021b) apply offensive language identification techniques and show that users with depression diagnosis use more offensive language in their Reddit posts, individuals manifesting signs of depression in their posts having a more profane language and fewer insults targeted towards other individuals or groups.

Computational models used to detect the cues of depression from social media texts rely on traditional machine learning classifiers (e.g. SVM, Naïve Bayes) (De Choudhury et al., 2013; Al-darwish and Ahmad, 2017; Tadesse et al., 2019), CNNs (Orabi et al., 2018; Yates et al., 2017), RNNs (Orabi et al., 2018) or transformer models (Martínez-Castañó et al., 2020; Uban and Rosso, 2020; Bucur et al., 2021a).

In the multimodal framework (involving text, voice and visual cues), the use of syntactic features (e.g. pronouns, adverbs usage) seems to improve the performance in depression detection, further emphasizing the relationship between linguistic features and depression (Morales et al., 2018).

Recently, researchers began exploring the interpretability and explainability of the decisions made by automatic classifiers for mental illnesses detection from social media to further understand the manifestations of different mental disorders in written language (Uban et al., 2021b,a).

3 Data

In our experiments, we use the dataset from the eRisk workshop containing posts written in English from the social media platform Reddit.

The eRisk 2018 dataset (Losada and Crestani, 2016) was created for the early detection of depression task. It contains two classes of users, depression and control. Users from the depression class were annotated by their mention of diagnosis in their posts (e.g. "I was diagnosed with depression"), but expressions such as "I have de-
pression”, “I am depressed” were not taken into account. The authors removed the mentions of diagnosis from the dataset. Users from the control group are random individuals who do not have any mention of diagnosis in their post, including those active in the depression subreddit\cite{Caragea2016}. The training dataset provided by the organizers contains 135 depressed users and 752 controls, while the test dataset contains 79 depressed users and 741 controls. We use both train and test splits in our exploration, consisting of a total of approximately 90,000 submissions from users with a depression diagnosis and over 985,000 in the control group.

4 Methods

Part-of-Speech Analysis For each post in our dataset, we use the spaCy\footnote{https://www.reddit.com/r/depression/} part-of-speech tagger to extract the corresponding tags and also morphological features (e.g. person and number for pronouns) for each word. We extract the universal POS tags\footnote{https://spacy.io/} and the ones from The Penn Treebank tagset\cite{Taylor2003}. We use the latter to explore the differences in the verb tenses. We assign the tenses according to their tags: VBD and VBN corresponding to past tense, VBG, VBZ and VBP corresponding to present tense, and MD tag before a VB corresponds to the future tense\cite{Caragea2016}. From the morphological features provided by the spaCy tagger, we extract the person and the number for all the pronouns. After this analysis, we use the following features in our exploration:

**Universal Part-of-Speech:** ADJ, ADV, NOUN, PROPN, VERB, ADP, CCONJ, DET, PART, SCONJ, AUX, PRON

**Verb tenses:** Past, present, future

**Person of pronouns:** First, second and third-person

**Pronoun number:** Singular and plural, only for the first-person

For each of these features, we compute their frequency for each post in the dataset. For the universal POS, the frequency is computed as the number of occurrences of a specific tag normalized by the total number of tags in a post. For verb tenses, the frequency of each tense is calculated as a percentage of the total number of verb occurrences. For the three kinds of personal pronouns, each frequency is computed as a percentage of the number of all personal pronouns. The frequency of singular and plural first-person pronouns is computed as a percentage of all first-person pronouns.

To further explore the part-of-speech usage in the social media dataset, we also use some special measurements\cite{Havigerova2019}:

**Pronominalisation Index (PI):** reflecting the usage of pronouns, instead of another part-of-speech (e.g. nouns). It is computed as the number of pronouns divided by the number of nouns\cite{Litvinova2016}.

**Formality Metric**\cite{Mairesse2007}: \( F_{\text{NOUN-ADJ-PREP-ART-PRON-VERB-ADV-INJ}} \)

Moreover, we test the discriminatory power of both POS tags frequency usage in users’ texts and the specific computed indices. For this, we employ a Random Forest classifier on the train set of the eRisk 2018 dataset on the aforementioned features. To interpret the trained model and to estimate the importance of each feature, we employ SHapley Additive exPlanations (SHAP)\cite{Lundberg2017} to measure each feature’s contribution to the classifier decision. SHAP offers a game-theoretic approach to quantify feature importance, aligned with human intuition.

**Classification** We opted for a simple Random Forest model, trained with 50 estimators and a max depth of 15, to avoid overfitting, with balanced class weights, since the dataset is heavily imbalanced. On the test set for the eRisk 2018 dataset, we obtain a weighted F\(_1\)-score of 78.37% (with balanced class weights) and a macro F\(_1\)-score of 51.93%. While the classification task is difficult, we are interested in exploring the feature importances of the model, which shed light on the model behaviour and provide us with insights regarding which POS tag is most discriminatory.

We further present our findings and provide interpretations and discussions based on recent findings in psychology.

5 Results and Discussion

Addressing our **RQ1**, we compute the Welch t-test for all our features and demonstrate that there are statistically significant differences (p-value <0.001) in part-of-speech usage between depressed and non-depressed individuals. In this
section, we present these differences and their interpretation from the psychology research.

Figure 1: Frequency of content part-of-speech

Content Part-of-speech  In Figure 1, we present the usage of content words for the two classes from the eRisk 2018 dataset. Individuals diagnosed with depression tend to use fewer common and proper nouns in comparison with control users. They also use more verbs and adverbs in their posts. The discourse is focused around actions, but with fewer entities (e.g. nouns), showing a defective linguistic structure with less interest in the environment (e.g. people, objects) (De Choudhury et al., 2016).

To further understand these differences, we pay a closer look at the frequencies of nouns and verbs in the social media discourse. We compute the keyness score (Kilgarriff, 2009; Gabrielatos, 2018) for verbs and nouns separately. In the keyness analysis, we compare the frequencies of nouns and verbs from the posts written by individuals with depression diagnosis (target corpus) in comparison to the posts from control users (reference corpus). In Figure 2, we present the top 20 verbs and nouns from each corpus, ordered by their log-likelihood ratio \( (G^2) \) (Dunning, 1993).

Rumination is a cognitive process focusing on past and present negative content and resulting in emotional distress (Sansone and Sansone, 2012). It is present in several mental health disorders (e.g. depression, anxiety, obsessive-compulsive disorder, post-traumatic stress disorder). In depression, rumination is defined as behaviors and thoughts that focus one’s attention on one’s depressive symptoms and on the implications of these symptoms (Nolen-Hoeksema, 1991). The rumination, as a response to depression, focuses the person’s attention on their emotional state and inhibits the actions necessary to distract them from their mood. In Figure 3, when comparing the usage of the three verb tenses (present, past and future) between the two groups, we expected that signs of rumination would be present in our analysis of verb tenses, with texts from depressed users being shifted into the past (Smirnova et al., 2018), but this result is not found in this sample of individuals.

Regarding the usage of future tense, depressed people have a lower frequency of verbs portraying future actions. This result may be a consequence of
anhedonia, people suffering from depression reporting lower anticipatory pleasure, and thus talking online less about their future plans. Anhedonia, defined as markedly diminished interest or pleasure in all, or almost all activities most of the day, nearly every day (Association et al., 2013), is a common symptom of depression.

The higher frequency of cognitive verbs (e.g. *feel*, *think*, *know*) in the texts written by depressed individuals indicates the cognitive impairments and judgement issues specific to depression. People with depression have cognitive deficits and biases in the processing of emotional information and they are unable to adaptively regulate their emotions (De Choudhury and De, 2014). Individuals with or without depression may not differ in their initial response to an adverse event. Still, they differ in their ability to recover once they have experienced the negative emotion. Depressed individuals are not able to repair their mood. Instead, they remain in a negative state of mind, which is related to increased negative thoughts, selective attention to negative stimuli and greater accessibility of negative recollections (Joormann, 2010). In comparison, the individuals from the control group use more action verbs (e.g. *vote*, *lead*, *show*, *begin*, *create*). In addition, depressed individuals are more passive; they have a lower level of general activity, consistent with symptoms of depressive disorder (Hopko et al., 2003).

Being an online social media platform similar to forums, Reddit is organized in subreddits with specific topics. It also has several communities dedicated to mental health problems. Compared to other social media platforms that require real-name authentication (e.g. Facebook), Reddit affords users anonymity or pseudo-anonymity. Complete anonymity is almost impossible, users providing bits of information with every interaction on the platform (e.g. comments, posts). Reddit allows users to create *throwaway* accounts to engage temporary without revealing their identity (Kilgo et al., 2018). These types of accounts are used to discuss sensitive information or stigmatizing problems. De Choudhury and De (2014) study the mental health discourse on Reddit and show that its communities allow a high degree of information exchange related to mental health. Users use Reddit to self-disclose the challenges faced in their daily lives or in personal relationships. They also seek emotional support or specific information about mental illnesses diagnosis and treatment. Their study demonstrates that Reddit fills the gap between social media platforms like Twitter and Facebook and online health forums regarding mental health discourse.

Examining the frequencies of nouns in the eRisk 2018 dataset, we show in Figure 2 that the users with self-reported depression diagnosis use their Reddit account to disclose and discuss their mental health problems (e.g. *depression*, *anxiety*, *therapist*) or their personal relationships (e.g. *friend*, *boyfriend*, *relationship*, *mom*, *dad*). The process of seeking online support is also shown in the frequency of verbs: *feel*, *talk*, *diagnose*, *help*.

Even if the dataset contains control users active in the depression subreddit, the majority of control users seem to post on other themed subreddits (e.g. politics). Bucur and Dinu (2020) perform a topic modelling analysis and show that texts from control users are found in topics related to their hobbies, as opposed to depressed people, who are more focused on their feelings and life events. Our results are in line with this study, the users from the control group use more politics-related words (e.g. *trump*, *government*, *president*, *news*, *vote*).

![Figure 4: Frequency of functional part-of-speech](image)

**Functional part-of-speech** In Figure 4, we present the frequencies of functional part-of-speech for depressed and control users. Depressed individuals generally use fewer functional words in their texts in comparison with control users, with the exception of pronouns. Neurons involved in content words processing are equally distributed over both hemispheres, while function words are processed mainly in the left hemisphere (Pulvermüller et al., 1995). Lower preposition usage may be an outcome of deficient activation of the left hemi-
sphere regions, responsible for producing more abstract lexical units (Litvinova et al., 2016). Function words are highly social, the capacity to use function words requires social skills (Pennebaker, 2017).

The Figure 5 shows the differences in personal pronouns used by the two groups. The high frequency of first-person singular pronouns indicates a higher self-preoccupation in depressed individuals, as opposed to the first-person plural pronouns, which shows collective attention. Second and third-person pronouns indicate social interactivity and contain references to other people or things in the environment (De Choudhury et al., 2016).

Depressed users use more first-person singular pronouns. The frequencies of first-person plural pronouns are inversely proportional to the first-person singular pronouns frequencies. This result is in line with the self-focused attention tendency (SFA) in depressed individuals. SFA is a cognitive bias linked to depression; the high frequency of first-person singular pronouns in speech or written text is considered a linguistic marker of SFA (Brockmeyer et al., 2015). Individuals with depression have deficits in other-focused social cognitions, they are impaired in Theory of Mind reasoning and empathy. Theory of Mind enables people to make inferences on the behaviour of others and their own (Premack and Woodruff, 1978). Erle et al. (2019) show that individuals exhibiting high levels of depressive symptoms were impaired on tasks involving overcoming their egocentrism.

The usage of fewer first-person plural pronouns in texts from users diagnosed with depression may be a sign of a lesser sense of belonging. The information-processing biases of depressed individuals make it hard for them to perceive cues of acceptance and belonging in social interactions, and to view ambiguous social interactions as being negative (Steger and Kashdan, 2009).
providing information about the context in order to avoid ambiguity (Heylighen and Dewaele, 2002).

Addressing our RQ2, we show in Figure 7 the Shapley values for a random sample of 1500 posts from the eRisk 2018 test set. A higher Shapley value corresponds with a higher importance in the final model decision based on the feature value. We used all computed POS features and indices in our model, but show only the top 20 for clarity. It is evident that from the summary plot, the absence of proper nouns is the most discriminatory factor in the decision to classify a person as depressive. Moreover, the use of pronouns (also evident in the Pronominalisation Index) is highly correlated with positive model output. The high usage of first-person singular pronouns and low usage of first-person plural pronouns confirms both our findings in the exploratory analysis and psychology literature.

6 Conclusion

In this work, we provide an extensive analysis of part-of-speech usage in social media texts from depressed and non-depressed users. Our findings are confirmed by studies in psychology and show that individuals diagnosed with depression use more pronouns (especially first-person singular pronouns) and verbs, and fewer common and proper nouns. Their social media discourse revolves around their life experiences and sentiments, as opposed to control users who are not interested in discussing their problems online.

Moreover, we also provided insights into the discriminatory power of POS frequencies by employing SHAP, a game-theoretic approach for model interpretation. Through this, we showed that depressive users can be characterized most easily, primarily through their usage of pronouns and proper nouns.

References


Michelle Morales, Stefan Scherer, and Rivka Levitan. 2018. A linguistically-informed fusion approach

Taleen Nalabandian and Molly Ireland. 2019. Depressed individuals use negative self-focused language when recalling recent interactions with close romantic partners but not family or friends. pages 62–73.


InFoBERT: Zero-Shot Approach to Natural Language Understanding
Using Contextualized Word Embedding

Pavel Burnyshev  Anfrey Bout  Valentin Malykh  Irina Piontkovskaya
Huawei Noah’s Ark lab, Moscow Russia
lastname.firstname@huawei.com

Abstract

Natural language understanding is an important task in modern dialogue systems. It becomes more important with the rapid extension of the dialogue systems’ functionality. In this work, we present an approach to zero-shot transfer learning for the tasks of intent classification and slot-filling based on pre-trained language models. We use deep contextualized models feeding them with utterances and natural language descriptions of user intents to get text embeddings. These embeddings then used by a small neural network to produce predictions for intent and slot probabilities. This architecture achieves new state-of-the-art results in two zero-shot scenarios. One is a single language new skill adaptation and another one is a cross-lingual adaptation.

1 Introduction

Dialogue systems become more and more popular in everyday life. A dialogue system has two main language understanding tasks which are of key importance for any skill: user intent recognition and information extraction from user input. The former task traditionally is interpreted as a classification problem, since the possible intents list assumed to be given. The latter task is formulated as slot filling, i.e. extraction of a text span considered to be a slot value. With dialogue systems becoming more popular their functionality is also growing in terms of both skills available to use and languages their functionality is accessible in. The rapid growth entails lack of usage data for the new skills and new languages and causes the lists of intents and slots to be dynamic thus make classical approaches inapplicable. We consider these newly emerged scenarios and demonstrate that our approach is well suited for both. In the first scenario, there are several known domains and a few unknown domains. The second scenario contains skills with training data for one language (English) and only test data for two other languages (Spanish & Thai).

In this paper, we present a new architecture which is based on pre-trained language models, like BERT (Devlin et al., 2019), and uses a small “head” (i.e. a neural network using the embeddings from the language models as input) evaluated in two scenarios: zero-shot single language natural language understanding on Schema-Guided Dialog dataset (Rastogi et al., 2019) and zero-shot cross-lingual natural language understanding on Multi-lingual Task-oriented Dialog dataset (Schuster et al., 2019). For the Schema-Guided Dialog dataset, there is a baseline described alongside the dataset which is evaluated in the zero-shot transfer learning setup. It is using deep contextualized embedding model (BERT) to embed the utterances and natural language descriptions for the slots and intents. Another approach was presented in (Ruan et al., 2020), where the authors present a system based on the baseline with an addition of hand-crafted context features. Our model is also close to this baseline with some important differences (described in section devoted to our approach) allowing it to outperform both previously presented approaches. For the Multi-language Task-oriented Dialog dataset there are also published baselines in zero-shot cross-lingual transfer setup described alongside the dataset. The authors train a whole machine translation system to make embeddings of utterances and use them for prediction in both tasks. Another approach proposed in (Liu et al., 2019b), where the authors use combination of recurrent neural network and latent variable model. Also an approach was presented in (Liu et al., 2020). The authors use BERT model to embed utterances and use special context gate for prediction in both tasks. We also use a pre-trained language-model to embed natural language descriptions of slots and intents and to achieve state of the art results in the
zero-shot scenario.

Formal contribution of this paper is four-fold: we present a new model architecture for (1) zero-shot intent classification and (2) zero-shot slot filling which with usage of appropriate embedding model achieves a new state of the art results on (3) cross-skill and cross-domain transfer and on (4) cross-language transfer.

2 Related Work

There are several baseline systems for MTD which we describe here, while baseline systems for SGD are described alongside our model in the specific section below. One baseline system (Multi. CoVe) presented in (Schuster et al., 2019) is based on deep contextualized embedding model CoVe (McCann et al., 2017). This system is using bidirectional LSTM (Hochreiter and Schmidhuber, 1997) over CoVe embeddings with CRF for slot prediction and an attention mechanism for intent recognition. This system has a variant with usage of an autoencoder for produced embedding refinement (Multi. CoVe w/ auto). (Liu et al., 2020) are using the same design as Multi. CoVe with other embedding models, namely multilingual BERT and XLM (Lample and Conneau, 2019), this approach is called Transformer-MLT. In this work is also presented adaptation of RCSLS model (Joulin et al., 2018) used as an embedding one, which aligns word embeddings with help of retrieval loss (RCSLS-MLT).

The authors of (Liu et al., 2019b) are using slightly different approach. Their system is using cross-lingually aligned word embeddings, they are fed into bidirectional LSTM, which output is used by attention mechanism for intent recognition and by a latent variable model for joint slot and intent prediction. In the work (Schuster et al., 2019) also presented another baseline called Translate Train. This system is using direct alignment of word tokens to fed this alignment into CRF for prediction.

There are models which use contextualized representations for natural language understanding in different setups. An approach to slot filling called Zero-Shot Adaptive Transfer described in (Lee and Jha, 2019). A model takes an utterance and natural language description of a slot and produces BIO-encoding of the utterance, i.e. points to the span, containing the value. To contextualize the word representations authors use bidirectional recurrent neural network. Another approach is presented in (Xu et al., 2020), where the authors use multilingual BERT as embedding model and predict jointly slots and intents with additional refinement task of data source prediction.

Deep contextualized embeddings, especially BERT, have been used for intent classification and slot filling tasks in other setups, e.g. (Chen et al., 2019; Chao and Lane, 2019; Zhang et al., 2019; He et al., 2020), where authors consider classic setup with shared slots and intents for train and test. There were also other works focused on zero-shot modeling (Bapna et al., 2017; Xia et al., 2018; Shah et al., 2019), domain adaptation and transfer learning techniques (Rastogi et al., 2017) in recent years. Deep learning based approaches have achieved state of the art performance on dialogue state tracking tasks. Popular approaches on small-scale datasets estimate the dialogue state as a distribution over all possible slot-values (Henderson et al., 2014; Wen et al., 2017) or individually score all slot-value combinations (Mrkšić et al., 2017; Zhong et al., 2018). Such approaches are not practical for deployment in virtual assistants operating over real-world services having a very large and dynamic set of possible values. Addressing these concerns, approaches utilizing a dynamic vocabulary of slot values have been proposed (Rastogi et al., 2018; Goel et al., 2019; Wu et al., 2019).

3 Task Description and Datasets

We consider two independent scenarios in this work, although they share an important feature: zero-shot transfer learning. The first scenario is a cross-skill and cross-domain transfer which is evaluated on Schema-Guided Dialog (SGD) dataset. It is formulated as follows: in each domain, there are one or more skills. Each skill has its intents and slots described. Also a skill could share an intent and/or a slot with another skill, but in general case, they are not sharing anything. To complicate things, in the dataset there is a domain (Alarm) which is presented only in the development set, but not in the train set.

The second scenario is a cross-lingual transfer and is evaluated on Multi-language Task-oriented Dialog dataset (MTD) dataset. Each skill presented in the dataset is described (to a reasonable extent) identically in all three languages. So a system could be trained on one language and evaluated on the other two languages.
Schema-Guided Dialog

Schema-Guided Dialog dataset is described in (Rastogi et al., 2019). This dataset contains task-oriented dialogues in different domains. Each skill has a so-called schema, which contains one-sentence descriptions of slots and intents used in this skill. Each dialogue has slots and intents marked up. It is important to mention that domain could be represented by more than one skill, e.g. a person could rent a car using two different services. The skill in one domain could be split into train and dev sets. Another important feature of this dataset is that it contains multi-domain conversations. The general statistics for this dataset is presented in Tab. 1. Additional statistics on the cross-domain dialogues distribution is presented in Tab. 2.

## Table 1: Summary statistics of the Schema-Guided Dialog dataset. Train/Dev values are separated with slash.

<table>
<thead>
<tr>
<th>Domain</th>
<th>#Intents</th>
<th>#Dialogs</th>
<th>Domain</th>
<th>#Intents</th>
<th>#Dialogs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm</td>
<td>2 (1)</td>
<td>37</td>
<td>Movie</td>
<td>4 (2)</td>
<td>1758</td>
</tr>
<tr>
<td>Bank</td>
<td>4 (2)</td>
<td>1021</td>
<td>Music</td>
<td>4 (2)</td>
<td>1486</td>
</tr>
<tr>
<td>Bus</td>
<td>4 (2)</td>
<td>2609</td>
<td>RentalCar</td>
<td>4 (2)</td>
<td>1966</td>
</tr>
<tr>
<td>Calendar</td>
<td>3 (1)</td>
<td>1602</td>
<td>Restaurant</td>
<td>4 (2)</td>
<td>2755</td>
</tr>
<tr>
<td>Event</td>
<td>5 (2)</td>
<td>3927</td>
<td>RideShare</td>
<td>2 (2)</td>
<td>1973</td>
</tr>
<tr>
<td>Flight</td>
<td>8 (3)</td>
<td>3138</td>
<td>Service</td>
<td>8 (4)</td>
<td>2090</td>
</tr>
<tr>
<td>Home</td>
<td>2 (1)</td>
<td>1027</td>
<td>Travel</td>
<td>1 (1)</td>
<td>2154</td>
</tr>
<tr>
<td>Hotel</td>
<td>8 (4)</td>
<td>3930</td>
<td>Weather</td>
<td>1 (1)</td>
<td>1308</td>
</tr>
<tr>
<td>Media</td>
<td>4 (2)</td>
<td>1292</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Multi-turn Task-oriented Dialog

Multi-turn Task-oriented Dialog dataset is described in (Schuster et al., 2019). This dataset consists of dialogues in three different languages, specifically English, Spanish, and Thai. The dialogues share semantics for intents and slots across the languages, which makes it possible to formulate a zero-shot cross-lingual task, i.e. a model could be trained on one language and evaluated on another language without any additional training. The semantics, in this case, is represented by one-sentence description for slots and intents. This dataset contains only one skill per domain and no multi-domain dialogues. The general statistics for this dataset is presented in Tab. 3. For this dataset, there is published train/validation/test split, which we follow in our experiments.

## Table 3: Summary statistics of the Multi-turn Task-oriented Dialog dataset. Note that the slot type `datetime` is shared across all three domains and therefore the total number of slot types is only 11. Train/Dev/Test values are separated with slash.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Utter-s</th>
<th>Slots</th>
<th>Dom-s</th>
<th>Intents</th>
<th>#Intents</th>
<th>#Dialogs</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>30.5/21/4.1</td>
<td>11</td>
<td>3</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spanish</td>
<td>11/18/8.621</td>
<td>11</td>
<td>3</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thai</td>
<td>2,156/1,235/1,692</td>
<td>3</td>
<td>3</td>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 4 Instruction Following BERT

We think of the natural language descriptions of intents and slots as instructions for a model to follow in order to achieve a result, which is either intent classification or slot filling. We base our approach on BERT-like models, especially their contextualization ability, and we show that these models could solve both parts with a small help of an additional simple module. We add to the pre-trained language-models a recurrence module for intent classification task and a feed-forward module for slot filling. We call our approach Instruction Following BERT (**InFoBERT**).

#### 4.1 BERT Model

To better present our work we start with brief description of the Bidirectional Encoder Representations from Transformers (BERT) model described in (Devlin et al., 2019). The modules in it are interconnected, so the model has access to the whole sequence at once. BERT model has a specific training procedure, which consists of two tasks: next sentence prediction and masked language modeling, that requires the model to use specific tokens, like [SEP] which separates different sentences in the text sequence, [CLS] which requires a model to make a decision (binary in the original setup), or [MASK] which hides a particular token from model input, so model is required use a context to perform a prediction of a masked token. The usage of these special tokens in training procedure is illustrated in Fig. 1. We have experimented with BERT and several derivative models, namely RoBERTa (Liu et al., 2019a) and ToD-BERT (Wu et al., 2019) for SGD dataset and
multilingual BERT, XLM (Lample and Conneau, 2019), XLM-RoBERTa (Conneau et al., 2020), and Language-Agnostic BERT Sentence Embedding (Feng et al., 2020) for MTD one.

Figure 1: BERT training setup. The figure adopted from (Devlin et al., 2019).

4.2 Intent Classification

For each skill, there is a known list of possible intents, and each intent is accompanied with a natural language description. To represent rejection in the classification we add special intent “NONE”, for which we use a separate description. This design is close to previous art for SGD. Although (Rastogi et al., 2019) system ignores a dialog history entirely, and (Ruan et al., 2020) system uses only hand-crafted history-based features. While our approach takes into account whole dialogue history using a recurrent neural network which is fed by a sequence of [CLS] token embeddings from all the utterances (including system ones). We denote [CLS] token embedding for \( i \)-th utterance as \( u_{i}^{\text{CLS}} \), and \( \hat{u}_{i}^{\text{CLS}} \) RNN output vector for this input. More formally:

\[
\{ \hat{u}_{i}^{\text{CLS}} \}_{i=1}^{n} = \text{RNN} \left( \{ u_{i}^{\text{CLS}}, c_{i-1} \}_{i=1}^{n} \right),
\]

where \( c_{i} \) is a state of the recurrent module on \( i \)-th step, and \( n \) is a number of utterances in a dialogue.

We denote \( I_{j}^{\text{CLS}} \) [CLS] token embedding from \( j \)-th intent description. To obtain the logits for possible intents, we compute dot-product between utterance vector representation and intent vector representation and normalize obtained scores using softmax (SM) function. More formally:

\[
P_{i}^{\text{intent}} = \text{SM} \left( \{ u_{i}^{\text{CLS}} \}^{T} \cdot I_{j}^{\text{CLS}} \mid j = 0..m \} \right),
\]

where \( m \) is a number of the intents in a skill, and \( I_{0} \) is reserved for “NONE” intent. The output logits are fed into conditional random field to incorporate further intent usage statistics.

To make the model more robust we use a Gaussian noise in form of \( d_{emb} \) normal distributions with zero mean and zero covariance adding it to all the embedding outputs of a language model. \( d_{emb} \) is an output language model embedding size.

4.3 Slot Filling

Generally, a slot has no list of possible values available, we call such slot the non-categorical one. In a case where the list of possible values is available, the slot is called categorical. MTD dataset contains only non-categorical slots, while SGD dataset contains both types. In this work we concentrate only on the non-categorical slots since they are more general and presented in both datasets we consider. The task is to extract a value from an utterance represented as a sequence of tokens, so the extracted value will be a span.

We denote \( s_{k}^{\text{CLS}} \) an embedding of [CLS] token for \( k \)-th slot description. Then to compute probability distributions for an utterance tokens (\( u_{il} \mid l = 1..L \)) to be a start (\( P_{i}^{k,\text{start}} \)) or an end (\( P_{i}^{k,\text{end}} \)) of a span we compute:

\[
P_{i}^{k,\text{start}} = \text{SM} \left( \{ u_{il}^{\text{CLS}} \} \cdot W_{\text{start}} \cdot s_{k}^{\text{CLS}} \mid l = 1..L \} \right),
\]

\[
P_{i}^{k,\text{end}} = \text{SM} \left( \{ u_{il}^{\text{CLS}} \} \cdot W_{\text{end}} \cdot s_{k}^{\text{CLS}} \mid l = 1..L \} \right),
\]

where \( L \) is an utterance length, \( W_{\text{start}} \) and \( W_{\text{end}} \) are trainable matrices. It is important to mention, that these matrices are the only additional weights we use to perform the task.

To train the model for non-categorical slots extraction we use classic cross-entropy loss, summing it over all possible slots and utterances.

To make the model more robust we add slot-value dropout. This technique is related to one described in (Xu and Sarikaya, 2014), but in our work we are replacing the whole value span with [MASK] tokens thus requiring the model to rely on the value context.

Our setup is close to previous art for SGD. Our approach differs on the one hand from (Ruan et al., 2020) with usage of additional matrices for start and end token prediction, and on the other hand from (Rastogi et al., 2019) with usage of only one matrix for each prediction, while the baseline is using two matrices. Both these approaches do not use slot-value dropout.

5 Experiments

We conduct experiments on two datasets, described in the section devoted to the datasets. For both SGD and MTD datasets we measure quality for tasks of intent classification and non-categorical slots extraction in the zero-shot scenario. To measure the intent classification quality, we use accuracy metric, due to a model has a dynamic list of intents to choose from. To measure the slot extraction quality we use F1 metric, which allows us to measure both
the precision and recall of a slot detection.

In our experiments we use LSTM as a unit of the recurrent module in the intent classification task. We use one layer with hidden size equal to the output embedding size of a underlying language model throughout all the setups.

In our experiments on the SGD dataset we measure quality only for intents and slots not present in the original training set. To conduct the experiments on the SGD dataset we use BERT (Devlin et al., 2019) as an embedding model, specifically BERT-base variant for both tasks (InFoBERT-B); and RoBERTa (Liu et al., 2019a) model, specifically RoBERTa-large for intent classification and RoBERTa-base for slot filling (InFoBERT-R). We also use ToD-BERT (Wu et al., 2020) model denoting it InFoBERT-T. For the slot filling task the slot-value dropout probability of 0.35 was used. Since there were no explicitly presented metrics we measure in the original SGD paper, we reproduce their results using publicly available code\(^1\). The results are presented in Tab. 4. As one could see from the table InFoBERT-T outperforms both the baselines in the combined domain intent recognition task. Unfortunately, (Ruan et al., 2020) did not presented results for single domain task, but basing on our experiments we expect the results for their model would be close to presented ones.

In our experiments on the MTD dataset we use English data as training set and Spanish & Thai data as a validation and test sets. For the MTD dataset we use multilingual BERT (Devlin et al., 2019) as an embedding model (InFoBERT-M), XLM (Lample and Conneau, 2019) model (InFoBERT-X), XLM-RoBERTa (Conneau et al., 2020) model (InFoBERT-XR), and Language-Agnostic BERT Sentence Embedding (Feng et al., 2020) model (InFoBERT-L). The noise standard deviation was set to 0.1 for InFoBERT-M model and to 0.01 for InFoBERT-X one. For the slot filling task the slot-value dropout was used with two probabilities 0.2 and 0.35. The results are presented in Tab. 5. In the table there are results named “Translate Train”. This is a special setup considered as a strong baseline. In this setup English data is translated to target language and used to train a model. These results and the results for Multi. CoVe [w/ auto] marked with * are adopted from (Schuster et al., 2019). The results of RCSLS-MLT and Transformer-MLT marked with † are adopted from (Liu et al., 2020).

As one could see from Tab. 5 InFoBERT-L variant of our model significantly outperforms all the baselines for Spanish language data, although for Thai language InFoBERT-XR is better than other baselines in intent prediction, with exception of Translate Train. For Thai slot tagging task InFoBERT-X variant is the best outperforming the Translate Train (and all the baselines).

Table 4: Schema-Guided Dialog dataset. Intent classification accuracy and Slot tagging F1 measure.

<table>
<thead>
<tr>
<th>Model</th>
<th>Single-dom.</th>
<th>Combined</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Int. Sl.</td>
<td>Int. Sl.</td>
</tr>
<tr>
<td>InFoBERT-B</td>
<td>0.890 0.899</td>
<td>0.879 0.903</td>
</tr>
<tr>
<td>InFoBERT-R</td>
<td>0.954 0.959</td>
<td>0.940 0.925</td>
</tr>
<tr>
<td>InFoBERT-T</td>
<td>0.982 0.965</td>
<td>0.955 0.967</td>
</tr>
<tr>
<td>(Ruan et al., 2020)</td>
<td>N/A N/A</td>
<td>0.948 0.983</td>
</tr>
<tr>
<td>(Rastogi et al., 2019)</td>
<td>0.748 0.883</td>
<td>0.773 0.891</td>
</tr>
</tbody>
</table>

Table 5: Multi-language Task-oriented Dialog dataset. Intent classification accuracy and Slot tagging F1 measure.

<table>
<thead>
<tr>
<th>Model</th>
<th>Spanish</th>
<th>Thai</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Int. Sl.</td>
<td>Int. Sl.</td>
</tr>
<tr>
<td>InFoBERT-M</td>
<td>76.10 66.28</td>
<td>48.40 6.75</td>
</tr>
<tr>
<td>InFoBERT-X</td>
<td>90.32 77.48</td>
<td>65.65 56.02</td>
</tr>
<tr>
<td>InFoBERT-XR</td>
<td>88.51 74.84</td>
<td>89.68 17.50</td>
</tr>
<tr>
<td>InFoBERT-L</td>
<td>96.65 86.74</td>
<td>61.79 13.84</td>
</tr>
<tr>
<td>RCSLS-MLT()†</td>
<td>87.05 59.12</td>
<td>81.44 30.42</td>
</tr>
<tr>
<td>Transformer-MLT()()†</td>
<td>87.88 74.88</td>
<td>73.46 28.47</td>
</tr>
<tr>
<td>(Liu et al., 2019b)</td>
<td>90.20 65.79</td>
<td>73.43 32.24</td>
</tr>
<tr>
<td>Multi. CoVe*</td>
<td>53.34 22.50</td>
<td>66.35 32.52</td>
</tr>
<tr>
<td>Multi. CoVe [w/ auto]*</td>
<td>53.89 19.25</td>
<td>70.70 35.62</td>
</tr>
<tr>
<td>Translate Train*</td>
<td>85.39 72.87</td>
<td>95.85 55.43</td>
</tr>
</tbody>
</table>

5.1 Thai Language Performance Analysis

The analysis of Tab. 5 shows that all the models with exception of InFoBERT-X variant of our model show significantly lower results on Thai slot filling task. The models we present and Transformer-MLT baseline could be split into two groups: the one using BERT tokenization model and the one using an external engine. We found out that all the models, including multilingual BERT, XLM-RoBERTa, Language-Agnostic BERT Sentence Embedding model, are using the same tokenization engine originally presented in BERT code\(^2\). Even XLM model by default is using this engine - this fact could explain low results for Transformer-MLT baseline. We found out that this
default engine is broken against Thai text, thus it corrupts input text during tokenization. But XLM model could be set to use external tokenization which handles the text correctly. This allows it to significantly improve the results on Thai language data.

It is interesting that the intent recognition results though being affected by this fault are still could achieve high performance. We suppose this fact is related to classification task structure where the whole utterance is regarded, allowing a model to overcome tokenization issues.

6 Conclusion

In this work we presented a model architecture which allows us to use different embedding models as a building block. This architecture is demonstrated to be effective in two zero-shot transfer tasks, namely cross-domain and cross-lingual. Our model using an appropriate embedding model (ToD-BERT for cross-domain task and several multi-lingual models for cross-lingual one) shows state of the art performance on the intent recognition and slot filling tasks.

It is interesting to mention that in our experiments we found that the best results for Spanish and Thai languages in the cross-lingual transfer task are not achievable at the same time. We suppose that this fact could be explained by the errors in the tokenization model used for most of the multi-lingual models. Thus usage of Language-Agnostic BERT Sentence Embedding model as an embedding one allows our system to outperform all the other systems on Spanish language data, but broken tokenization in this model does not allow to show any improvement for Thai language.

As future work authors consider the study of other BERT descendant models, which are plenty nowadays. Another direction of the work is research in low resource scenario, when some data is available for a model to tune onto. In closing, we hope that this work will facilitate the research in the direction of transfer learning in dialogue systems.

References


Active Learning for Assisted Corpus Construction: A Case Study in Knowledge Discovery from Biomedical Text

Hian Cañizares-Díaz¹, Alejandro Piad-Morffis¹, Suilan Estevéz-Velarde¹, Yoan Gutiérrez², Yudivian Almeida Cruz¹, Andres Montoyo², and Rafael Muñoz-Guillena²

¹School of Math and Computer Science, University of Havana, Cuba
{hian.canizares,apiad,sestevez,estevanell,yudivian}@matcom.uh.cu
²University of Alicante, Spain
{ygutierrez,monoyo,rafael}@dlsi.ua.es

Abstract

This paper presents an active learning approach that aims to reduce the human effort required during the annotation of natural language corpora composed of entities and semantic relations. Our approach assists human annotators by intelligently selecting the most informative sentences to annotate and then pre-annotating them with a few highly accurate entities and semantic relations. We define an uncertainty-based query strategy with a weighted density factor, using similarity metrics based on sentence embeddings. As a case study, we evaluate our approach via simulation in a biomedical corpus and estimate the potential reduction in total annotation time. Experimental results suggest that the query strategy reduces by between 35% and 40% the number of sentences that must be manually annotated to develop systems able to reach a target $F_1$ score, while the pre-annotation strategy produces an additional 24% reduction in the total annotation time. Overall, our preliminary experiments suggest that as much as 60% of the annotation time could be saved while producing corpora that have the same usefulness for training machine learning algorithms. An open-source computational tool that implements the aforementioned strategies is presented and published online for the research community.

1 Introduction

Machine learning, and specifically supervised learning, is one of the most effective tools for automating complex cognitive tasks, such as recognising objects in images or understanding natural language text. One of the main bottlenecks of supervised learning is the need for high-quality datasets of labelled samples on which statistical models can be trained. These datasets are usually built by human experts in a lengthy and costly manual process. Active learning (Cohn, 2010) is an alternative paradigm to conventional supervised learning that has been proposed to reduce the costs involved in manual annotation.

The key idea underlying active learning is that a learning algorithm can perform better with less training examples if it is allowed to actively select which examples to learn from (Settles, 2009). In the supervised learning context, this paradigm changes the role of the human expert. In conventional supervised learning contexts, the human expert guides the learning process by providing a large dataset of labelled examples. However, in active learning the active role is shifted to the algorithm and the human expert becomes an oracle, participating in a labelling-training-query loop. In the active paradigm, a model is incrementally built by training on a partial collection of samples and then selecting one or more unlabelled samples to query the human oracle for labels and increase the training set. This approach introduces the new problem of how to best select the query samples so as to maximise the model’s performance while minimising the effort of the human participant.

The simplest active learning scenario consists of the classification of independent elements $x_i$ drawn from a pool of unlabelled samples. Examples range from image classification (Gal et al., 2017) to sentiment mining (Kranjc et al., 2015), in which the minimal level of sampling (e.g., an image or text document) corresponds to the minimal level of decision. i.e, a single label is assigned to each $x_i$. More complex scenarios arise when the decision level is more fine-grained than the sampling level. In the domain of text mining, an interesting scenario is the...
The task of entity and relation extraction from natural language text (Zhang et al., 2012). In this scenario the sampling level is a sentence, but the minimal level of decision involves each token or pair of tokens in the sentence, and furthermore, these decisions are in general not independent within the same sentence. In this case, it is not trivial to estimate how informative an unlabelled sample will be, since each sample has several sources of uncertainty.

This research focuses on the tasks of entity and relation extraction, proposing an active learning strategy to reduce the overall time of annotation for human experts, by actively selecting the most informative sentences to annotate. We also consider the problem of providing some instances of entities and relations pre-annotated to further reduce the annotation time for the human, while minimising the number of erroneous suggestions. In contrast with the usual formulation, we focus on the problem of obtaining the corpus per-se, and the task of training the underlying machine learning models is considered as means to an end rather than as the primary objective.

The contributions of this research can be summarised as follows:

- **We present an active learning strategy for the problem of entity and relation extraction from natural language text that greatly reduces the annotation time for human experts by actively selecting the most informative sentences and providing pre-annotated suggestions when possible.**

- **We propose an informativeness measure for entity and relation extraction that factors in the uncertainty of annotations in a sentence counter-balanced by its similarity to the labelled set.**

- **We evaluate our proposal in an experimental corpus in the biomedical domain, and study the impact of the query strategy and the benefit of providing pre-annotated suggestions, in terms of reducing the overall time of annotation.**

- **As a tangible result, we provide the source code for a prototype annotation tool that implements the aforementioned strategies and is compatible with the BRAT annotation tool, available online under an open-source license (https://github.com/knowledge-learning/assisted-annotation).**

The remaining sections of this paper are organised as follows. Section 2 reviews the most relevant research related with active learning in general and specifically for entity and relation extraction. Section 3 presents the formal definition for our active learning approach. Section 4 describes a computational prototype tool that implements this strategy. Section 5 evaluates our proposal in the context of a corpus of entities and relations in the biomedical domain. Section 6 presents a discussion of the most relevant insights that our research provides. Finally, Section 7 presents the main conclusions of our research.

2 Related Works

This section reviews some of the most relevant research related with active learning in general, and specifically focused on entity detection and relation extraction. One of the most important design decisions in active learning is how to intelligently select the novel unlabelled samples in the most efficient way. The underlying assumption is that we want to train a model to the highest possible performance (measured in precision, $F_1$, etc.) while minimising the human cost (measured in time, number of samples manually labelled, or any other suitable metric). This requirement is often framed as the selection of the most informative unlabelled samples, and formalised in terms of a query strategy (Settles, 2009). The most common query strategies for general-purpose active learning can be grouped into the following categories:

(i) **Uncertainty sampling**: The most informative samples are considered those with the highest degree of uncertainty, given some measure of uncertainty for each sample (Lewis and Catlett, 1994).

(ii) **Query by committee**: The most informative samples are considered those with the highest disagreement among a committee of either different models or different hypotheses from the same underlying model (Seung et al., 1992).

(iii) **Expected model change**: The most informative samples are considered those that produce the highest change in the model’s hypothesis...
if they were included in the training set (Settles et al., 2008).

(iv) Variance and error reduction: The most informative samples are those which produce the highest reduction in the model’s generalisation error or, as a proxy, its variance (Roy and McCallum, 2001).

Expected model change (iii) and variance/error reduction (iv) strategies are heavily dependent on the specific learning model used. In contrast, uncertainty sampling (i) and query by committee (ii) are applicable in general with a high degree of model agnosticism. Furthermore, relevant subsets of both strategies can be formalised under a single framework if we define the uncertainty as a measure of the entropy of the model’s predicted output. In this framework, query-by-committee can be implemented via weighted voting, thereby assigning empirical probabilities to the possible outputs.

Weighted density is a complimentary strategy in which the most informative samples are weighted by how representative they are of the input space, for example, by measuring their similarity to the remaining samples (Settles and Craven, 2008). This approach attempts to counter-balance a noticeable tendency to select outliers as the most informative samples—a problem associated with other query strategies—since outliers are often the samples that create the highest amount of uncertainty, disagreement or hypothesis change.

Recent advances in natural language processing have produced an increased interest in active learning to alleviate the requirement for large annotated corpora (Olsson, 2009; Tchoua et al., 2019). Settles and Craven (2008) compare several strategies for active learning in sequence labelling scenarios, concluding that query strategies based on measures of sequence entropy combined with weighted sampling outperform other variants. Meduri et al. (2020) propose a comprehensive benchmark to evaluate different active learning strategies for entity matching. In the task of named entity recognition, CRF models have been used to select query samples (Claveau and Kijak, 2017; Lin et al., 2019). The task of relation extraction also benefits from active learning approaches, both in general-purpose settings (Fu and Grishman, 2013) and in domain-specific settings (Zhang et al., 2012). However, despite the growing body of research, it is still a challenge to apply active learning in joint entity recognition and relation extraction, especially in scenarios with low resources (Gao et al., 2019).

3 Active Learning Strategy for Entity-Relation Annotation

This section presents our active learning strategy for human-in-the-loop annotation of corpora based on entity recognition and relation extraction. A high-level overview of the process is illustrated in Figure 1.

Our active learning strategy is designed for an arbitrary corpora of independent natural language sentences, each of which must be annotated by a human expert at token level. We consider a pre-defined set \( E \) of entity labels, each of which can span one or more tokens, continuous or discontinuous. Additionally, there is a predefined set \( R \) of binary relation types between entities, where the possible relations between each pair of entities can depend on the entity type, i.e., not all relation types are defined for all entity labels. There is no sub-token annotation, and not all tokens need to be annotated. This abstract annotation schema can represent a broad range of different tasks, from domain-specific relation extraction (e.g., gene-protein interaction) to general-purpose semantic representation (e.g., AMR parsing).

The active learning strategy proposed in this research works iteratively in batches of \( K \) sentences (e.g., \( K = 10 \)). At any point there will be a labelled pool \( L \) with \( |L| = n \times K \) sentences that have been manually annotated by a human annotator, and a large unlabelled pool \( U \) of raw sentences. Initially, the human annotator selects \( K \) representative sentences and performs a full manual annotation (step 0). Afterwards, two machine learning models are iteratively trained on the manually labelled sentences (step 1) and a metric of informativeness, \( I(s) \), is computed for each sentence \( s \in U \) (step 2). The top \( K \) sentences in terms of \( I(\cdot) \) are selected (step 3) and the model produces a prediction of entity and relation labels for each one (step 4). Each prediction has an associated metric of uncertainty, \( H(\cdot) \), estimated by the models. Based on this uncertainty and pre-defined thresholds \( u_e \) and \( u_r \) for entities and relations respectively, all the entities \( e_i \) (relations \( r_j \)) with an estimated uncertainty \( H(e_i) > u_e \) (\( H(r_j) > u_r \)) are discarded. Finally the selected and partially annotated sentences are presented to the human annotator, who must correct the incorrect annotations.

223
and add the missing ones (step 5). The corrected sentences are incorporated to the labelled pool for the next iteration (step 6).

The following components for the active learning strategy need to be specified: a machine learning model $M_E$ that predicts entity labels; a machine learning model $M_R$ that predicts relations; and, suitable definitions for $I(\cdot)$ and $H(\cdot)$ based on these machine learning models. We will not define specific machine learning models at this point, since different models can be suitable for different corpora. For our strategy to work, the machine learning models $M_E$ and $M_R$ are only required to provide a probability distribution over the possible labels rather than a single prediction. This is a soft restriction that many machine learning models comply with.

To measure the informativeness $I(s_i)$ of each sentence $s_i \in U$, we define a metric based on uncertainty sampling with weighted density, inspired by Settles and Craven (2008).

First, given the set of $n$ entity annotations $E_i \in E^n = \{ e^i_1, \ldots, e^i_n \}$ and $m$ relation annotations $R_i \in R^m = \{ r^i_1, \ldots, r^i_m \}$ predicted for a sentence $s_i$, we define the uncertainty of each entity $e^i_k$ (or relation $r^i_k$) as the entropy of the probability distribution for all possible labels for that entity or relation. Formally:

\[
H(e^i_k) = -\sum_{l_j \in E} P(e^i_k = l_j | s_i; \theta) \log_2 P(e^i_k = l_j | s_i; \theta)
\]

\[
H(r^i_k) = -\sum_{l_j \in R} P(r^i_k = l_j | s_i; \theta) \log_2 P(r^i_k = l_j | s_i; \theta)
\]

Where $\theta$ represents the parameters of the machine learning model used to estimate these probabilities.

We can define the mean uncertainty associated to the predicted entities and relations, respectively, as follows:

\[
\bar{H}(E_i) = \frac{1}{n} \sum_{e^i_k \in E_i} H(e^i_k)
\]

\[
\bar{H}(R_i) = \frac{1}{m} \sum_{r^i_k \in R_i} H(r^i_k)
\]

Second, we define an information density metric $ID(s_i)$ to estimate how representative each sentence $s_i$ is with respect to the input space. In a similar formulation to Settles and Craven (2008), $ID(s_i)$ is defined as the average similarity of the sentence $s_i$ to the cluster of $K$ labeled sentences:

\[
ID(s_i) = \frac{1}{K} \sum_{s_j \in L_i^*} \text{sim}(s_i, s_j)
\]

Where $L_i^*$ is the subset of $K$ labeled sentences that maximize the similarity metric with respect to $s_i$. Any sensible similarity metric can be used. In this research we propose the use of Doc2Vec embeddings (Le and Mikolov, 2014) pre-trained on the unlabeled set $U$ to estimate sentence similarity.
Finally, the overall informativeness of an unlabeled sentence \(s_i\) is estimated based on the uncertainty measures \(\hat{H}(\cdot)\) of each component, weighted by the information density of the sentence:

\[
I(s_i) = \left[ \hat{H}(E_i) + \hat{H}(R_i) \right] \times ID(s_i)^\beta
\]

Where \(\beta\) is a scaling factor to balance exploitation versus exploration, i.e., decreasing the uncertainty of the model versus selecting more varied sentences to reduce model bias.

If we consider the annotation of a sentence as a stochastic process, where each entity or relation annotation is a random event, then \(\hat{H}(\cdot)\) is a finite approximation of the entropy rate of the annotation process. This provides an intuitive interpretation for the informativeness measure \(I(\cdot)\) in the domain of information theory. The most informative sentences are those whose entropy rate is maximum (weighted by density). Maximum entropy rate has been successfully applied to feature selection in other machine learning scenarios (Einicke et al., 2017).

4 Computational Prototype

The strategy presented in Section 3 is implemented as a web application that can be integrated with the BRAT annotation tool (Stenetorp et al., 2012). This application is independent of BRAT and relies only on the file system to iteratively select batches of sentences and apply suggestions. The web interface is simple to use, allowing the user to ask for a new batch, and decide whether to accept, modify or discard the annotation suggestions (see Figure 2). This tool is compatible with any entity and relation annotation schema that can be represented in BRAT Standof ANN format (Yepes et al., 2013).

As explained in Section 3, two different machine learning models \(M_E\) and \(M_R\) must be implemented to evaluate the informativeness metric \(I(\cdot)\). These models must provide probability estimates for each label, and should be efficient enough to be trained in the same time it takes a human annotator to annotate a single batch, such that the new batch is always ready. For the previous reasons, we selected two simple machine learning models based on standard formulations for the problems of entity recognition and relation extraction respectively.

For entity model \(M_E\), we select a conditional random field (CRF) classification model with syntactic and semantic features extracted with the spacy library. The extracted features include coarse and fine-grained part-of-speech tagging, lemmatization, a standard NER labeling, as well as indicator variables for several syntactic patterns (e.g., numbers, dates, punctuation, emails, URLs, etc.). By this means, the entity recognition problem is framed as a sequence tagging problem using the BILOUV encoding and Viterbi decoding. Special hand-crafted rules are designed to account for multi-word entities with discontinuous word spans. The uncertainty of each entity \(H(e_k)\) is estimated by the normalized marginal probabilities of the CRF model on the token sequence, averaging the probabilities of the tokens that correspond to the same entity. Despite its simplicity, this model achieves an \(F_1\) score of 0.78 in the entity extraction subtask of the eHealth-KD Challenge 2020, which is competitive with state-of-the-art techniques in past benchmarks (Piad-Morffis et al., 2019b).

In the case of the relation model \(M_R\), this subtask is more complex and simple baselines perform significantly worse than state-of-the-art models. However, since complex models cannot be trained in the required time, we decided to maintain a simple baseline. The problem of extracting all relations in a sentence is modeled as a set of independent classification problems between all pairs of entities in the sentence. Each pair is represented by the same characteristics used in the entity recognition subtask, applied to both entities under analysis, plus a bag-of-words encoding of the tokens that appear in the smallest dependency subtree that contains both entities. The uncertainty \(H(r_{ij})\) of each pair is computed from the marginal probabilities provided by a logistic regression model trained on each pair representation. For the information density metric \(ID(\cdot)\) an implementation of Doc2Vec from the gensim library is used.

5 Experiments and Results

To validate the effectiveness of the active learning strategy proposed in this research we selected a recent manually annotated corpus of Spanish sentences in the biomedical domain, i.e., the eHealth-KD 2020 corpus (Piad-Morffis et al., 2020). This selection was motivated by the relative complexity of the annotation schema proposed in this corpus, which contains different entity and relation types, multi-word tokens, overlapping annotations and other characteristics that make it a challenging annotation process even for human experts (Piad-Morffis et al., 2019a). The corpus contains a total
Figure 2: Screenshot of the web application prototype for semi-automatic corpus annotations integrated with the BRAT annotation tool. The right panel shows an illustrative selection of annotated sentences in the schema of the eHealth-KD 2020 corpus (Piad-Morffis et al., 2020), see Section 5.

of 1300 sentences in Spanish, manually annotated with a general-purpose entity-relation schema. Of these, a set of 1000 sentences is used as the unlabeled pool \(U\), and the remaining 300 are used for testing the final performance of all machine learning models trained in the experiment. The corpus has been split following the authors’ recommendations. Figure 2 shows an illustrative example of the annotation schema applied to 3 exemplary English sentences, in the context of the prototype application developed in this research.

We simulated the assisted annotation process to evaluate the effect to annotating the corpus using active learning strategies versus annotating the corpus in the original order without suggestions (baseline). As the process of annotating a corpus is expensive, it was simulated using the gold annotations in the training collection. The improvement can be estimated by comparing how many sentences need annotating to reach a specific performance of the machine learning algorithms (measured in terms of \(F_1\) in the testing collection).

To study the relative impact of the different components of our query strategy, we evaluated three different variants. They consisted of using the full query strategy proposed in Section 3 with \(\beta = 1\), as well as considering only entity uncertainty \(\hat{H}(E_i)\) and relation uncertainty \(\hat{H}(R_i)\) respectively. Figure 3 shows how the \(F_1\) metric improved with each batch of sentences, for the first 500 sentences. The target \(F_1\) is the final score obtained by training the models \(M_E\) and \(M_R\) on the full 1000 sentences of the corpus. In general, the curves that correspond to the active learning strategy (i.e., assisted variants) approach the target \(F_1\) significantly faster than the unassisted baseline.

To illustrate the degree of time-reduction achieved, Figure 4 shows the minimum number of sentences that must be annotated to reach different relative target \(F_1\) scores. For example, after annotating the first 400 sentences it is possible to achieve a 95% of the ultimate \(F_1\) score when using all the corpus. However, to reach the target score, the first 880 out of a 1000 sentences must be annotated if the corpus is annotated in the original order (baseline). By contrast, using our active learning strategy only between 530 to 580 sentences must be annotated to reach the same target \(F_1\), thereby saving between a 35% and a 40% of human annotation time.

Another interesting finding is to estimate the extent to which the suggested annotations further reduce the total annotation time. A human annotator using the tool will need to accept some of the suggested annotations, correct the ones that are wrong and annotate the ones that are missing. Each of these actions has a different cost in time. For quantifying the improvement in overall time that the suggestions produce, we assigned a relative cost (in terms of abstract time units) to each of the following types of annotations:

**Missing annotations:** annotations that the model did not suggest and the human annotator must produce, have a cost of 1 time unit.

**Spurious annotations:** annotations that the model suggested and are wrong, which must be eliminated by the human annotator, have a cost of 2 time units.

**Correct annotations:** since the human annotator must at least recognise the annotation is correct, the cost is 0.25 time units.
Partial annotations: annotations that are partially correct either because the spans are partially covering or the label is wrong, have a cost of 0.5 time units.

This cost structure assumes that the problem of correcting wrong annotations is more complex than simply producing the correct annotations, while acknowledging that even agreeing with correct annotations has a non-zero cost. For an active learning strategy to be helpful it must provide enough correct annotations to outweigh the cost of correcting the wrong annotations; hence, it should prioritise precision over recall.

Figure 5 shows the relative effect (in terms of reducing the overall annotation time) of enabling annotation suggestions for different combinations of the entity and relation thresholds \( u_e \) and \( u_r \). It can be observed that on average, the entity suggestions produce a positive effect (green colour) across a wide range of thresholds, while the relation suggestions tend to produce a negative effect as more suggestions are allowed. This is a direct consequence of the \( M_R \) relation model’s performance, which achieves at most an \( F_1 \) score of 0.27, while the entity model \( M_E \) achieves up to a 0.78 score. The optimal time reduction is achieved for an entity threshold \( u_e = 2.4 \) and a relation threshold \( u_r = 0 \), producing an estimated 24% reduction in the total annotation time. Interestingly, these parameters result in an overall performance for the machine learning model of \( F_1 = 0.54 \), with a precision of 0.78 and a recall of 0.41. As expected, given the asymmetrical cost structure, it is preferable to prioritise precision rather than recall for the annotation suggestions.

An overall reduction in annotation time for this experimental simulation can be estimated by combining the improvements provided by annotation suggestions and the sentence ordering. Assuming both effects are independent, the best case scenario for this corpus suggests the following. Using the active learning approach a human annotator would have needed to annotate only 530 sentences out of 1000, each of them with an estimate time cost of 76% compared to the full annotation. This results in an overall reduction of as much as 60% of the total annotation time, producing a smaller corpus on which machine learning models can still be trained, delivering the same performance as those trained on the original corpus.

6 Discussion and Future Work

The machine learning model used for entity recognition \( M_E \) achieves a result comparable with the state-of-the-art in this corpus while being simple enough to be trained during annotation. Not only does the model produce a significant reduction in the number of sentences that need to be annotated but it also is capable of pre-annotating entities that are often correct, even when factoring in the significantly higher cost of correcting the wrong suggestions. By contrast, the relation extraction model \( M_R \) performed significantly worse than current state-of-the-art in this corpus. However, even if the pre-annotated relations suggested by this model are on average not beneficial, it is interesting to note that using only the uncertainty of relations \( H(R_i) \) as a query strategy still produces a significant time reduction (see Figure 4, assisted (relations)). Un-
Unfortunately, all good performing models for this problem are composed of complex deep learning architectures that cannot be trained sufficiently fast enough to be used during the annotation process in a commodity hardware.

Indeed, using simple models is necessary in active learning scenarios where algorithms must be trained interactively, but even without considering this issue, there are additional factors to consider related to model complexity. We argue that an interesting trade-off exists between the capacity of a model and its usefulness for active learning. Very simple models (underfit) will have a high uncertainty in all samples, while very complex models (overfit) will overestimate their certainty. In both cases, the informativeness \( I(\cdot) \) for all sentences will be very similar, and there is no sensible way to choose the most informative ones. This suggests that there may be an optimal middle ground where the model learns enough to provide useful suggestions while still maintaining a healthy level of uncertainty. The fact that even weak baselines (like the relation model \( M_R \)) are still a useful source of information when actively selecting unlabelled sentences is one surprising conclusion of our research. This seems to suggest that even in very complex scenarios where state-of-the-art models are impossible to train interactively, using weaker surrogate models can still provide a significant benefit for human-in-the-loop learning.

Regarding the generalisation of our approach, the fact that the corpus is in the Spanish language is irrelevant for our experimental results since the machine learning models used are language-agnostic and no language-specific heuristics were applied. Hence, these results should generalise to other languages and annotation schemas albeit with different baseline \( F_1 \) scores according to the complexity of the underlying learning problem. Nevertheless, we are interested in evaluating our approach using languages other than English since the creation of linguistic resources is one of the main difficulties of NLP research, especially for other languages. An ongoing research priority is to validate this strategy on other corpora with different annotation schemas.

In future work, we will explore how to explicitly control the complexity of a model during the active learning process by controlling the model’s capacity. Two strategies that can be analysed are the use of ensemble methods and deep learning architectures with early stopping. In both cases, the intuitive idea is to iteratively refine a machine learning model up to the point where a sufficiently good performance is achieved but before the model overfits on the small labelled set of sentences, such that uncertainty measures are still relevant. Another interesting scenario in which to apply this approach is when many annotators exist for the same text. In this case, the models can learn contradictory hypotheses due to differences between annotators. Interestingly, in the case of active learning, this is a positive phenomena, since inter-annotator disagreement is a meaningful proxy for annotation difficulty. Active learning models trained on a pool of sentences with multiple, possibly contradictory annotations, will naturally tend to select sentences that are more likely to cause disagreement between annotators. In this context, it can be interesting to explore query-by-ensemble methods where each model in the ensemble is trained on a different annotator’s pool to maximise model variance.

7 Conclusions

In this article, we present an approach for reducing the time involved in manually annotating a corpus of natural language sentences that contains entities and relations. This approach uses active learning with uncertainty sampling and weighted density, and provides an estimated reduction of 60% of total annotation time in a simulated experiment with a real corpus. This improvement is derived from two independent factors: intelligently sorting which...
sentences to annotate and providing pre-annotated suggestions with a high-degree of certainty. The proposed strategies have been implemented into a computational tool that is applicable to a broad range of corpus annotation schemas and is available for the research community.

Acknowledgments

This research has been supported by a Carolina Foundation grant in agreement with University of Alicante and University of Havana. Moreover, the research has been partially funded by the University of Alicante and the University of Havana, the Generalitat Valenciana (Conselleria d’Educació, Investigació, Cultura i Esport) and the Spanish Government through the projects LIVING-LANG (RTI2018-094653-B-C22), INTEGER (RTI2018-094649-B-I00) and SIIA (PROMETEO/2018/089, PROMETEO/2018/089). Additionally, it has been backed by the work of both COST Actions: CA19134 – Distributed Knowledge Graphs and CA19142 – Leading Platform for European Citizens, Industries, Academia and Policymakers in Media Accessibility.

References


Alejandro Piad-Morfis, Yoan Gutiérrez, Juan Pablo Consuegra-Ayala, Suilan Estevez-Velarde, Yudivián Almeida-Cruz, Rubio Muñoz, and Andrés Montoyo. 2019b. Overview of the eHealth Knowledge Discovery Challenge at IberLEF 2019. In IberLEF@SEPLN.


Multiple-Instance Active Learning. In J. C. Platt, 


Unsupervised Text Style Transfer with Content Embeddings

Keith Carlson
Department of Computer Science
Dartmouth College
Hanover, NH 03755
keith.e.carlson.gr@dartmouth.edu

Allen Riddell
Department of Information and Library Science
Indiana University Bloomington
Bloomington, IN 47405

Daniel Rockmore
Departments of Computer Science and Mathematics
Dartmouth College
Hanover, NH 03755
& The Santa Fe Institute
Santa Fe, NM 87501

Abstract

The style transfer task (here style is used in a broad “authorial” sense with many aspects including register, sentence structure, and vocabulary choice) takes text input and rewrites it in a specified target style preserving the meaning, but altering the style of the source text to match that of the target. Much of the existing research on this task depends on the use of parallel datasets. In this work we employ recent results in unsupervised cross-lingual language modeling (XLM) and machine translation to effect style transfer while treating the input data as unaligned. First, we show that adding “content embeddings” to the XLM which capture human-specified groupings of subject matter can improve performance over the baseline model. Evaluation of style transfer has often relied on metrics designed for machine translation which have received criticism of their suitability for this task. As a second contribution, we propose the use of a suite of classical stylometrics as a useful complement for evaluation. We select a few such measures and include these in the analysis of our results.

1 Introduction

In this paper we consider the problem of unsupervised holistic textual style transfer – both the accomplishment of the task, as well as its evaluation. The “style” of text is roughly the way in which a text communicates its content. It might be thought of as the “voice” characteristic of a given author, an emergent quality that encompasses a wide range of (more or less measurable) characteristics such as register, sentence structure, and vocabulary choice. Holistic style transfer takes a given text – written a priori in one “style” – and then rewrites it (preserving its content) in another style. Holistic style transfer is distinct from more narrow style modification techniques which manipulate specific characteristics of prose such as formality, simplicity, or sentiment.

To illustrate our idea of holistic style consider the following pair of translations of the opening lines of The Aeneid of Virgil.

Arms and the man I sing, the first who came, Compelled by fate, an exile out of Troy...(Humphries et al., 1987)

I sing of arms and the man who of old from the coasts of Troy came, an exile of fate, to Italy and the shore of Lavinium...(Mackail, 1885)

As another example compare verses from two different “versions” of a fixed verse from the Book of Genesis, the first from the King James Version

And a river went out of Eden to water the garden; and from thence it was parted, and became into four heads.

and the second, the same verse, but in the New International Version:

A river watering the garden flowed from Eden; from there it was separated into four headwaters.

In both example pairs we can see that the content in the passages is the same, but the (holistic) style differs noticeably.

The examples above are effectively examples of a human-executed style transfer. The potential applications of a machine holistic style transfer are numerous. For example, various periodicals often try to have a single “voice” and an unsupervised
style transfer of the kind studied here would enable a staff writer to produce the content required of an article which was then “stylized” per the requirements of the venue. Thus, a style transfer platform could be a high-powered editorial assistant. Such a platform could also assist aspiring writers. All that said, one should not be blind to the more nefarious potential of successful style transfer machinery which could be useful for spoofing an audience to productive, or unproductive writerly ends (Nature, 2020).

One machine learning approach to holistic style transfer is to adopt and adapt the frameworks of translation models, treating each style as a language. Along these lines, much of the existing research on this task depends on the use of parallel datasets, a schema that follows early work in machine translation, but parallel datasets in this domain are in fact rare. This motivates our approach wherein we continue to be inspired by machine translation work and employ recent results in unsupervised cross-lingual language modeling (XLM) to effect holistic style transfer while treating the input data as unaligned, an important next step in advancing this area in light of the scarcity of parallel datasets. Additionally, we show that modifications to this framework which take advantage of the differences between the style transfer and machine translation tasks can improve model performance. Specifically, we add “content embeddings” to the XLM which capture human-specified groupings of subject matter and observe improvement over the baseline model for a range of metrics.

That brings us to the paired challenges resident in evaluating a style transfer technique. This task is complicated by the emergent nature of style. The analogy of style transfer to translation and concomitant efforts to use techniques from machine translation for the style transfer task have inspired the importation of evaluation metrics from machine translation to the style transfer setting, (Xu et al., 2012; Jhamtani et al., 2017; Carlson et al., 2018), although not without criticism (Tikhonov et al., 2019; Xu et al., 2016). As the evaluation of a style transfer task should a priori measure the similarities of source and target texts to their “native environments”, it seems natural to bring to bear some of the techniques from the field of stylometry, a discipline focused on the quantitative analysis of textual style. Stylometry (or stylometrics) was born of a nineteenth century effort to settle – quantitatively – scholarly dispute around the temporal ordering of Plato’s Dialogues (Lutoslawski, 1897)). For this task, over 500 individual and measurable textual characteristics were identified. Since that time stylometrics have been used (most famously) to address questions of disputed authorship (see e.g., Mosteller and Wallace (1964); Boyd and Pennebaker (2015)). If we imagine a system which perfectly performs style transfer as we have defined it, then the output of the system – in terms of its individual characteristics – should be stylistically indistinguishable from text written by the author whose style is targeted. It thus seems natural to use a range of stylometric measures used in the past to distinguish between authors’ styles as an evaluation for the performance of such a system. This line of reasoning motivates a second contribution of this work wherein we introduce the idea of using stylometric measures for evaluation. We evaluate our systems using several stylometric measures in addition to the more commonly previously used metrics and show that the stylometric approach is a useful domain specific complement to translation-based metrics for the evaluation of the complex, subtle, and important task of style transfer.

2 Related Work

Style transfer has some long roots. It is possible to frame the early work on text simplification (e.g., Specia (2010)) or paraphrasing Xu et al. (2012) as a form of style transfer. Style transfer research makes use of a range of datasets for training and evaluation. Examples include the corpus of Shakespearean plays and their “translations” into contemporary English (Xu et al., 2012) for paraphrasing and a corpus of Wikipedia pages and their simplified versions (Zhu et al., 2010) which is used for the general task of text simplification.

The more stylistic features that are incorporated into building a model, the closer it gets to the kind of holistic effort we have described above. To that end, we highlight Ficler and Goldberg (2017) wherein a supervised style transfer model is developed which focuses on the modification of prose with respect to six aspects of style, including register, sentiment, focalisation, and prolixity. A broader approach for supervised holistic style transfer is addressed in Carlson et al. (2018); Xu et al. (2012). They make use of a model that depends on a corpus of versions of the Bible, a priori aligned through the canonical and shared structuring of
Book, Chapter, and verse, to learn the differences between examples written in different styles.

Unsupervised methods pose new challenges for style transfer. Previous related work uses unsupervised training for generating text in a particular style. This includes the generation of stylized text (Hu et al., 2017) and modification of the sentiment or formality of prose (Shen et al., 2017; Li et al., 2018; Gong et al., 2019; Li et al., 2019). There have also been advances in the use of unsupervised approaches for machine translation. Many of these rely on the idea of back-translation (Artetxe et al., 2017; Lample et al., 2018) to automatically generate a synthetic parallel from unaligned data. Lample and Conneau (2019) uses this concept along with a novel cross-lingual language model objective for pre-training to achieve impressive performance on the unsupervised translation task.

3 Experiments

3.1 Data

Our work makes use of eight cleaned and aligned public domain versions of the Bible introduced in Carlson et al. (2018) and made available on Github. (That paper mentions the availability of thirty-four versions, but twenty-six of them have copyright restrictions that restrict their distribution.) These represent eight different English writing styles. The texts are divided hierarchically (and canonically), into version, book, chapter and verse, so that the verses from different versions are parallel. For our unsupervised work we do not take advantage of the alignment during training, but the alignment does enable an objective evaluation of our output.

Our major methodological advance is the introduction of another coarse level of hierarchy which we call content, which we use to modify the language model. We see this kind of coarse labeling as an approach which is broadly generalizable to situations in which fine-scaled parallel alignment does not exist. In the case of the Bible, we use nine “divisions” of the Bible which are classical groupings of thematically similar texts.1 See Table 1 for the divisions used. We do not use the exact data splits detailed in Carlson et al. (2018), but instead split the data as required by the formulation of our models. We use some books of the YLT (Young’s Literal Translation) and BBE (Bible in Basic English) versions for validation and testing as style transfer between these versions was identified as the “hardest” task in Carlson et al. (2018). The validation set contains the BBE and YLT versions of 1 Kings, Zephaniah, Mark, and Colossians. The testing set contains the BBE and YLT versions of Judges, 1 Samuel, Philippians, and Hebrews. The remaining books from BBE and YLT and all books from the other six (publicly available) Bible versions make up the training data.

The parallel texts allow for automatic and objective evaluation of translations. While the models we describe can be generalized to other non-parallel datasets, in those cases objective evaluation would be more difficult.

<table>
<thead>
<tr>
<th>Division</th>
<th>Books</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pentateuch</td>
<td>Genesis, Exodus, Leviticus, Numbers, Deuteronomy</td>
</tr>
<tr>
<td>History</td>
<td>Joshua, Judges, Ruth, 1 Samuel, 2 Samuel, 1 Kings, 2 Kings, 1 Chronicles, 2 Chronicles, Ezra, Nehemiah, Esther</td>
</tr>
<tr>
<td>Poetry</td>
<td>Job, Psalms, Proverbs, Ecclesiastes, Song of Solomon</td>
</tr>
<tr>
<td>Major Prophets</td>
<td>Isaiah, Jeremiah, Lamentations, Ezekiel, Daniel</td>
</tr>
<tr>
<td>Minor Prophets</td>
<td>Hosea, Joel, Amos, Obadiah, Jonah, Micah, Nahum, Habakkuk, Zephaniah, Haggai, Zechariah, Malachi</td>
</tr>
<tr>
<td>(Pauline) Epistles</td>
<td>Romans, 1 Corinthians, 2 Corinthians, Galatians, Ephesians, Philippians, Colossians, 1 Thessalonians, 2 Thessalonians, 1 Timothy, 2 Timothy, Titus, Philemon, Hebrews</td>
</tr>
<tr>
<td>General Epistles</td>
<td>James, 1 Peter, 2 Peter, 1 John, 2 John, 3 John, Jude</td>
</tr>
<tr>
<td>Revelation</td>
<td>Revelation</td>
</tr>
</tbody>
</table>

Table 1: Our partition of Bible books into divisions.

3.2 Baseline System

Lample and Conneau (2019) introduced a method for cross-lingual language model pretraining from non-parallel data2. Their model, XLM, feeds token, position, and language embeddings to a Transformer model (Vaswani et al., 2017) which tries to predict masked words. This task, Masked Language Modeling (MLM), was introduced by Devlin et al. (2018) and unsupervised translation was

---

1There is no authoritative partition into divisions, but there are many similar varieties. Our choice among these options is somewhat arbitrary, but has historical and disciplinary support. An example of Old Testament divisions which match ours can be found at [http://www.scriptureman.com/ot.gif](http://www.scriptureman.com/ot.gif) and our New Testament at [http://jpatton.bellevue.edu/inspired-table2.jpg](http://jpatton.bellevue.edu/inspired-table2.jpg)

2Code found at: [https://github.com/facebookresearch/XLM](https://github.com/facebookresearch/XLM)
demonstrated as an application of these pretrained language models. We use the XLM as our baseline.

In our experiment, we treat each version of the Bible in the data as a language. So the embeddings fed to the Transformer for MLM training are position and token embeddings as before, and version embeddings replacing the language embeddings of the original system. Our transformer architecture has embeddings of length 512, 6 layers, 8 attention heads and a 0.1 dropout rate.

We train the language model from scratch until the perplexity of the validation data for the BBE→YLT version has stopped improving. We then use this pretrained language model to initialize Transformers for both the encoder and decoder of our machine translation/style transfer model and train on the task of unsupervised translation until the BLEU score of the validation data for the BBE→YLT task has stopped improving. This design is based on that used by Lample and Conneau (2019) in the original paper. We call these models “XLM”.

3.3 Model with Content Embeddings

Using Bible divisions as a grouping of content similarity, we modify the XLM embedding structure accordingly and include a content embedding in addition to the token, position, and language (style) embeddings. In a different context other considerations or structural organization may suggest a different articulation of content. This additional embedding is treated similarly to the three embeddings in the baseline system. The input of each token passed to the Transformer is the combination of four embeddings instead of three. Just as in the XLM, these embeddings are updated during the training process. Our intuition is that for some datasets, the model may have difficulty distinguishing whether differences in language arise because of differences in the style of writing, or differences in the content. By providing training data where both style and content are designated, we anticipate that the model will be better able to reproduce the differences which are style-specific. Similar intuition has led to other approaches which allow a model to learn style and content separately (Fu et al., 2017; Zhang et al., 2018).

In this new formulation, we provide all four embeddings to the Transformer and then train towards the MLM objective as before. We call this model “XLM + Content” (see Figure 1). We use the same parameter settings as in the “XLM” model and as before, we stop training of the language model when the perplexity of BBE→YLT evaluation task has stopped improving. Once again this transformer which was pretrained on the MLM task is used to initialize the encoder and decoder of a machine translation/style transfer model. This transfer model continues training until the BLEU score of the evaluation data BBE→YLT has stopped improving. Note that the alignment (parallel nature of the texts) makes possible the BLEU scoring.

![Figure 1: “XLM + Content” model training on the MLM objective. Based on Figure 1 of Lample and Conneau (2019). The choice of types for content embeddings are human-assigned before training as seen in Table 1.](image)

4 Results

4.1 Evaluation Metrics

The existence of parallel texts allows us to evaluate our results using the standard translation quality measures BLEU (Papineni et al., 2002) and PINC (Chen and Dolan, 2011), which reward similarity to the target and dissimilarity to the source respectively. PINC was created from a desire to “measure lexical dissimilarity with the source sentence” and its creators say “In essence, it is the inverse of BLEU” (Chen and Dolan, 2011). The results of these evaluations can be seen in Table 2.

We find that our model with content embeddings has a higher (better) PINC score for all four test books, indicating that it has more aggressively made changes than the baseline system. “XLM + Content” also attains a sizeably Higher BLEU score on Philippians and Hebrews. The BLEU score for the other two test books are similar between the two systems.
Table 2: The BLEU (PINC) scores of the unmodified source and the output of each model for each test book. All scores are when translating from Bible version YLT to Bible version BBE.

<table>
<thead>
<tr>
<th>Test Book</th>
<th>Source</th>
<th>XLM</th>
<th>XLM+Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Judges</td>
<td>16.18(0)</td>
<td>26.5(39.93)</td>
<td>26.1(44.89)</td>
</tr>
<tr>
<td>1 Samuel</td>
<td>14.75(0)</td>
<td>24.21(39.72)</td>
<td>24.36(44.40)</td>
</tr>
<tr>
<td>Philippians</td>
<td>18.29(0)</td>
<td>20.56(25.50)</td>
<td>22.82(29.83)</td>
</tr>
<tr>
<td>Hebrews</td>
<td>12.27(0)</td>
<td>15.88(29.73)</td>
<td>17.44(34.70)</td>
</tr>
</tbody>
</table>

4.1.1 Stylometry-Inspired Evaluation

This combination of BLEU and PINC scores for evaluating style transfer in text has been used in other work (Xu et al., 2012; Jhamtani et al., 2017; Carlson et al., 2018), but not without criticism (Tikhonov et al., 2019; Xu et al., 2016). Arguably, style transfer – especially for the situation in which there is no parallel (aligned) text – cries out for new kinds of measures. As mentioned in the Introduction, we believe that classical stylometric measures provide a natural source of appropriate options. Some approaches to stylometry are structural, while others focus on word usage frequency. For example, function word-based approaches have proved to be a useful (partial) fingerprint for authorial style in some cases (see e.g., (Mosteller and Wallace, 1964; Binongo, 2003)).

Thus inspired we augment the use of BLEU and PINC through several stylometrically inspired metrics. The first is the identification of frequent idiosyncratic words, words that seem simultaneously characteristic of one style but absent or rare in another. This form of bespoke evaluation checks to see if 17 frequent words with known translations have been correctly translated in the YLT→BBE test task. All the words occur frequently and exclusively in YLT. Examples include unto, hath, flee, doth and the full list can be seen in Table 3. These words occur 2,522 times in YLT source lines in the test set. In this test, a YLT→BBE translation is counted as correct if the BBE version does not include the idiosyncratic word from the YLT line. Accuracy scores in this evaluation increase with the complexity of the model: 99.3% (“XLM”) and 99.8% (“XLM + Content”).

In addition to this test of frequent idiosyncratic words, we analyze the entire test set of source, reference, and model outputs with a few other simple stylometrics: number of multi-syllable words, average number of syllables per word, average number of letters per word, and number of complex words (Dale and Chall, 1948). The results can be seen in Table 4. On all 4 of these evaluation metrics we find that the model modified to include separate content embeddings (XLM+Content) is closer to the target BBE than is the unmodified XLM model. This analysis provides further evidence that the content embeddings are enabling the model to produce better results.

4.2 Example Outputs

Table 5 shows two test data example inputs and their targets alongside the corresponding outputs of our systems. In the first example, note that both outputs correctly remove the use of quotation marks as is consistent with the BBE target and modernize the archaic Thou and dost. The “XLM + Content” also correctly changes the word testify to witness. In the second example, the “XLM + Content” model correctly changes age-during to eternal.

5 Conclusion and Future Work

The task of holistic textual style transfer requires a system to take text in a native (source) style as input and then rewrite the text, retaining the meaning while changing the style consistent with a specified target. In many potential applications this task will need to be performed in contexts where there is no parallel data which captures the styles of interest available for training. Examples range from the journalistic (writing articles in a given editorial style) to the literary (writing the style or voice of a given author). Contexts such as these have large corpora of source and target examples, but presumably – no source/target pairings.

In this work we demonstrated that a modern unsupervised machine translation technique could be applied to unsupervised holistic textual style transfer in the context of different styles (well known and publicly available versions) of the Bible. We
Table 4: Comparison of simple stylometric measures on whole test set.

<table>
<thead>
<tr>
<th></th>
<th>YLT(Source)</th>
<th>BBE(Ref)</th>
<th>XLM</th>
<th>XLM + Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-Syllable Words</td>
<td>3596</td>
<td>2231</td>
<td>2597</td>
<td>2369</td>
</tr>
<tr>
<td>Syllables Per Word</td>
<td>1.31</td>
<td>1.23</td>
<td>1.25</td>
<td>1.24</td>
</tr>
<tr>
<td>Letters Per Word</td>
<td>3.96</td>
<td>3.75</td>
<td>3.8</td>
<td>3.75</td>
</tr>
<tr>
<td>Complex Words</td>
<td>13189</td>
<td>7967</td>
<td>9480</td>
<td>8946</td>
</tr>
</tbody>
</table>

Table 5: Examples Outputs of each of the systems with YLT source and BBE target.

<table>
<thead>
<tr>
<th></th>
<th>YLT(Source)</th>
<th>BBE(Ref)</th>
<th>XLM</th>
<th>XLM + Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 and one in a certain place did testify fully, saying, ‘What is man, that Thou art mindful of him, or a son of man, that Thou dost look after him?</td>
<td>6 But a certain writer has given his witness, saying, What is man, that you keep him in mind? what is the son of man, that you take him into account?</td>
<td>6 And one in a certain place did testify fully, saying, What is man, that you are mindful of him, or a son of man, that you do look after him?</td>
<td>6 And one in a certain place did give witness fully, saying, What is man, that you are mindful of him, or a son of man, that you will go after him?</td>
<td></td>
</tr>
<tr>
<td>9 and having been made perfect, he did become to all those obeying him a cause of salvation age-lasting.</td>
<td>9 And when he had been made complete, he became the giver of eternal salvation to all those who are under his orders;</td>
<td>9 And having been made perfect, he did become to all those who obey him a cause of salvation age-lasting.</td>
<td>9 And having been made perfect, he gave to all those who keep him a cause of salvation eternal,</td>
<td></td>
</tr>
</tbody>
</table>

show that by adding an additional “content embedding layer” to encode the type of content in text, holistic style transfer is improved. The parallel nature of Bible versions enables us to objectively measure the effect of our innovation of content embedding – improvement is witnessed in terms PINC and BLEU scores that are greater when using content embedding than when not. Specifically, this improves upon the work of Carlson et al. (2018) and makes use of their publicly accessible data. We further introduce new measures of style transfer quality (a simple test of frequent idiosyncratic words as well as source/target comparisons of some basic stylometric measures – number of multi-syllable words, syllables per word, letters per word, number of complex words) as novel evaluations of style transfer, supplementing the traditional – and by some accounts, somewhat flawed – use of the PINC and BLEU metrics in this context. These new measures are a contribution in their own right to the space of evaluation frameworks for style transfer and also support our claim that content embedding improves style transfer.

The broader range of possible stylometric evaluation measures suggests that at least with respect to evaluation, a requirement of perfect evaluation and parallel texts might be relaxed.

While the Bible may seem to be particularly suited to the partition into content classes we employ, we believe this technique can be directly applied to many other textual sources as well. Similar to Bible versions, many translations exist of other classical works such as the epics written by Homer or Dante. In many of these translations alignment does not exist line by line so traditional supervised methods are not applicable. They are however “softly aligned” by book or chapter making content embeddings a natural choice. A model trained on these could then produce Homer’s Iliad in the style of a translator who only produced a version of the Odyssey. Similarly, many translations of classic non-English novels exist and this system could be used to create a new translation targeting the style of a particular translator.

Demand for English-to-English style transfer also exists commercially. Examples here include poetry parodies (Zaranka, 1981), continuations of stories from famous authors (James, 2011), or modernized retellings of stories (Rivers, 2012; McKinley, 2011). In these cases the content of the text either exists publicly or is written by the author. The style however is intentionally changed, either to match the works of another writer, or to remove the idiosyncrasies of the original style. Unsuper-
vised style transfer models could be used to help produce these works.

In addition to these potential applications, our results reinforce the idea that consideration of content and style independently can improve the results of style transfer models. In cases where our technique cannot be directly applied, this provides additional evidence to researchers that finding a way to separate the two may improve results.

In conclusion, this work highlights the utility of the Bible as a dataset for holistic style transfer, demonstrates that unsupervised machine translation methods for holistic style transfer are possible and can be objectively evaluated, provides further evidence — and an actionable methodology — for the idea that learning content independent of style can be beneficial, and proposes the use of classical stylometric measures for evaluation of style transfer systems.

References


Frederick Mosteller and David L Wallace. 1964. Inference and Disputed Authorship: The Federalist. Addison-Wesley, Reading, MA.


Evaluating Recognizing Question Entailment Methods for a Portuguese Community Question-Answering System about Diabetes Mellitus

Thiago Castro Ferreira\(^1\) João Victor de Pinho Costa\(^1\) Isabela Rigotto\(^1\) Vitoria Portella\(^1\) Gabriel Frota\(^1\) Ana Luisa A. R. Guimarães\(^1\) Adalberto Penna\(^1\) Isabela Lee\(^1\) Tayane A. Soares\(^1\) Sophia Rolim\(^1\) Rossana Cunha\(^1\) Celso França\(^1\) Ariel Santos\(^1\) Rivane F. Oliveira\(^1\) Abisague Langbehn\(^2\) Daniel Hasan Dalip\(^3\) Marcos André Gonçalves\(^1\) Rodrigo Bastos Fóscolo\(^1\) and Adriana Pagano\(^1\)

\(^1\)Federal University of Minas Gerais (UFMG), Brazil
\(^2\)Technical High School of UFMG, Brazil
\(^3\)Federal Center for Technological Education of Minas Gerais (CEFET-MG), Brazil

thiagocf05@ufmg.br

Abstract

This study describes the development of a Portuguese Community-Question Answering benchmark in the domain of Diabetes Mellitus using a Recognizing Question Entailment (RQE) approach. Given a premise question, RQE aims to retrieve semantically similar, already answered, archived questions. We build a new Portuguese benchmark corpus with 785 pairs between premise questions and archived answered questions marked with relevance judgments by medical experts. Based on the benchmark corpus, we leveraged and evaluated several RQE approaches ranging from traditional information retrieval methods to novel large pre-trained language models and ensemble techniques using learn-to-rank approaches. Our experimental results show that a supervised transformer-based method trained with multiple languages and for multiple tasks (MUSE) outperforms the alternatives. Our results also show that ensembles of methods (stacking) as well as a traditional (light) information retrieval method (BM25) can produce competitive results. Finally, among the tested strategies, those that exploit only the question (not the answer), provide the best effectiveness-efficiency trade-off. Code is publicly available\(^1\).

1 Introduction

Question answering (QA) aims to automatically retrieve precise, rather than merely relevant, answers to a given question. The field has faced exponential progress along the years with new corpora (Rajpurkar et al., 2016; Ahmad et al., 2019) as well as computational models which approach the task from different perspectives. One of these is known as Recognizing Question Entailment (RQE).

Given a premise question (aka query), a RQE approach aims to retrieve semantically similar archived questions which have been already answered (Ben Abacha and Demner-Fushman, 2019). The task became relevant thanks to popular Community QA forums, such as Yahoo Answers, Quora and Stack Overflow, where an RQE approach is used to automatically search a large body of material looking up for archived question-answer pairs entailing a user posed question.

Besides popular forums, a domain in which RQE approaches are highly beneficial and have been extensively studied is the medical one. Russell-Rose and Chamberlain (2017) showed that, when using traditional information retrieval search engines to query medical information, healthcare professionals spend on average 60 minutes to formulate a search strategy, 3 minutes to analyse the relevance of a retrieved document, and 4 hours of total search time. Ben Abacha and Demner-Fushman (2019) suggest that healthcare consumers may also benefit from QA systems through which they can ask for desired information in natural language instead of having to perform complex search strategies. In fact, a study reveals that, in 2013, 59% of U.S. adults searched for health information online and 35% used healthcare search engines to figure out what medical condition they or someone else had\(^2\).

To advance the state-of-the-art in RQE approaches for medical-specific applications, some benchmarks have been proposed. Targeting Frequently Asked Question (FAQ) by healthcare consumers, Abacha and Demner-Fushman (2016) introduced a collection of 8,890 pairs of questions labelled as having or not the same meaning. In the TREC 2017 LiveQA track, a medical question answering task was proposed addressing the automatic answering of consumer health questions received by the U.S. National Library of Medicine (Abacha et al., 2017). Under the shared-task, a

\(^1\)https://github.com/Dia-Bete/RANLP2021

\(^2\)https://www.pewresearch.org/internet/2013/01/15/health-online-2013/
total of 738 question-answer pairs were publicly released. Finally, Ben Abacha and Demner-Fushman (2019) released MedQuad, a dataset with 47,457 medical question-answer pairs of 37 question types extracted from 12 websites of the National Institutes of Health of United States.

Although the RQE task has been extensively studied, as is the case with other NLP subfields, studies and approaches largely focus on the English language. In this study, we go in a different direction, facing the challenge of developing an NLP approach for a low resource language. Specifically, our study focuses on the Portuguese language and investigates the development of a Community Question Answering system in the particular domain of Diabetes Mellitus using RQE. This allows investigating new challenges in terms of multi-lingual NLP problems.

From an application domain (Health) perspective, world-wide, the number of people with diabetes increased from 108 million in 1980 to 422 million in 2014\(^3\). It is expected that the disease will reach 629 million in 2045. These numbers are partially explained by the rapidly increase of the disease in low- and middle-income countries, such as Brazil, where around 12 million people have this health condition. Aiming to cope with this disease and to have a better quality of life, a significant number of Portuguese speakers with this condition engages in dedicated public forums. In order to improve the access to information about this health condition for these speakers, this study aims to leverage RQE approaches to build BeteQA, a Portuguese community question-answering system to provide prompt and accurate answers to questions about Diabetes Mellitus posed by social forum users.

To meet our goal, similar to Abacha and Demner-Fushman (2016); Abacha et al. (2017) and Ben Abacha and Demner-Fushman (2019), we first built a Portuguese benchmark with 785 pairs between premise questions and archived answered questions annotated as perfect match, relevant and irrelevant, following Nakov et al. (2016, 2017). Based on our benchmark corpus, we leveraged and evaluated several RQE approaches ranging from traditional information retrieval (IR) methods (BM25 and TF-IDF cosine similarity) to novel large pre-trained language models such as BERT (Devlin et al., 2019).

For the latter methods, we evaluated them according to both zero-shot and fine-tuned learning setups. We also used ensemble models (stacking) based on a learn-to-rank model to combine the outputs of the previous methods with other traditional linguistic features.

Experimental results show that a supervised transformer-based method trained in multiple languages and for multiple tasks (MUSE) outperforms the alternatives in a zero-shot setting. Moreover, results show that the ensemble method (stacking) as well as a traditional (light) IR method (BM25) have the potential to provide competitive results. Finally, among the tested strategies, those that exploit only the question (not the answer), provide the best effectiveness-efficiency trade-off. Our failure case analysis also reveals that most failures occur for longer sentences containing a smaller number of relevant candidates and harder separability.

The remainder of this paper is organized as follows: Section 2 describes how the benchmark was built. Section 3 explains the RQE models leveraged to rank similar questions about Diabetes. Section 4 describes the experimental methods while Section 5 discusses the experiment results and a failure analysis. Section 6 presents related work and Section 7 concludes our study.

### 2 Data

**Collection** To build a Portuguese RQE benchmark in the domain of Diabetes Mellitus, we first manually extracted Portuguese questions about this health condition from specialized Websites and Social Media forums. In particular, most questions were extracted from the FAQ section in the website of the Brazilian Association about Diabetes\(^4\) as well as in forums about this health condition, such as DIABETES - DIABÉTICOS\(^5\), a Facebook community about Diabetes with around 85 thousand Portuguese speaking users.

**Preprocessing and Anonymization** To keep users’ privacy, the extracted questions from forums were manually de-identified by first removing emojis, fixing orthographic mistakes and paraphrasing non-fluent syntactic structures which could point to idiosyncratic wordings by users. Then any identifier, such as name, phone or address, was removed from the questions. Moreover, quasi-identifiers,
such as age and relative mentions, were modified. Users’ age were modified by randomly choosing a number in the interval of \([\text{age} - 5; \text{age} + 5]\), whereas mentions to relatives were randomly changed by the reference to a relative with similar age, such as parent ↔ uncle, parent in law; sibling ↔ cousin, partner; soon ↔ nephew. We collected a total of 1474 questions.

**Answers**

4 Medical students were recruited to review the question-answer pairs extracted from FAQ sections of websites as well as to formulate answers to the de-identified questions from public online forums. During the evaluation process, each question was answered by a single medical student. In case of doubts, the respective question would be discussed among the students together with a medical specialist. To organize the process, the students kept a standardized database of answers to the collected questions. Each answer in this database was classified according to 10 topics: 1) General information about Diabetes; 2) Diagnosis; 3) Chronic complications; 4) Acute complications; 5) Treatment; 6) Treatment control; 7) Comorbidities; 8) Signs and symptoms; 9) Motivation; and 10) Highly frequent, though unrelated to diabetes.

Answers were elaborated pursuant to Article 37 of Chapter V of the Brazilian Code of Medical Ethics, which prohibits treatment prescription without actual patient examination. Hence, the answers were constructed with the aim of informing the user about diabetes and related issues, without offering any diagnosis or treatment. In cases where the user requested some type of intervention, answers were prepared in order to guide them to seek a public healthcare unit, both to obtain an accurate diagnosis and to have adequate therapeutic plans designed by healthcare professionals.

**RQE benchmark**

To finally build the benchmark, we randomly selected 200 questions (roughly 15%) as premises, whereas the remaining 1274 (together with their answers) were indexed by a BM25 model (Jones et al., 2000). For each premise question, we retrieved the 5 most similar candidate questions, together with their answers, using BM25. Finally, following (Nakov et al., 2016, 2017), given 1000 triples (premise question, candidate question, candidate answer), a medical student was recruited to annotate whether the candidate question was a perfect match, relevant or irrelevant to the premise one. The candidate question was considered a perfect match when it conveyed exactly the same semantic meaning as the premise question. When both candidate and premise questions shared the same topic, but were not semantically identical, the candidate was labeled as relevant. Otherwise, the candidate question was labeled as irrelevant to the premise question.

Once the annotation was concluded, premise questions with only irrelevant candidate questions were ruled out, resulting in a corpus with 157 premise questions, each one aligned with 5 annotated question-answer pairs.

**3 Models**

Drawing on our collected Portuguese benchmark about Diabetes, we evaluated several approaches to rank question-answer pairs to their premise questions. Such approaches range from traditional bag-of-words information retrieval techniques to novel methods based on continuous vector representations and Learn-to-rank ensembles.

**3.1 Token-Based Approaches**

**BM25** is a fast information retrieval technique (Jones et al., 2000) which, in the context of RQE, calculates the relevance of archived question-answer pairs to a given premise question using a family of scoring functions based on bag-of-words.

**Cosine Similarity over TF-IDF** ranks the similarity of archived documents, such as questions or question-answer pairs, to the premise question by computing the cosine similarity between their TF-IDF vector representations (Salton and McGill, 1986). TF-IDF is a bag-of-words technique, standing for *term frequency–inverse document frequency*. As the name implies, a TF-IDF vector representation of a document is computed by counting the frequency of its tokens as well as their specificity, defined by an inverse function of the number of documents in which each of its tokens occurs.

**3.2 Embedding-Based Approaches**

Currently, sparse bag-of-words vector representations have given place to dense vectors computed by neural networks and popularly known as (word, sentence or document) *embeddings* (Mikolov et al., 2013). We leveraged some of these representations as RQE approaches.
3.2.1 Skip-Gram Wang2Vec
We used the Brazilian Portuguese word embeddings of 300 dimensions computed by a skip-gram Wang2Vec architecture described in Hartmann et al. (2017). To obtain the embedding representations of a multi-word document such as a question, we first looked up for the word-embeddings of each of its tokens and then averaged them. Absent tokens in the skip-gram Wang2Vec vocabulary were represented by the embedding of the OOV (out-of-vocabulary) token. During ranking, we used cosine similarity to measure the semantic distance between a premise question and its candidates.

Limitation Skip-Gram Wang2Vec provides context-free representations of words, i.e. the approach does not distinguish the meaning of a particular occurrence of a word taking its surrounding context into account. For instance, the word *bank* would be represented by the same vector representation in the expressions “financial bank” and “river bank”.

3.2.2 BERT-Based Approaches
More recently, several studies have proposed large neural language models which compute context-sensitive word embeddings representations (Howard and Ruder, 2018; Peters et al., 2018; Devlin et al., 2019). These language models take the local word context into account to generate its meaning representation. One of the first and most popular approaches of this kind is the “Bidirectional Encoder Representations from Transformers” (BERT) (Devlin et al., 2019), which encodes the meaning of a word into a vector taking its surrounding words (before and after) into account. BERT is pretrained in an unsupervised fashion using objective functions like word-denoising (i.e., predicting a masked word in a text) and next sentence prediction. In our study, we leveraged some of its multilingual and Brazilian Portuguese variations to the RQE task in the Diabetes Mellitus domain.

**BioBERTpt** is another variation of BERT which was pre-trained with Brazilian Portuguese Clinical texts (Schneider et al., 2020) for the task of named entity recognition in the target domain.

3.2.3 MUSE
Differently from BERT that trains large language models in an unsupervised fashion style, other studies have sought to learn a semantic vector space among words and documents by training a neural network in a supervised context with multiple downstream tasks. A state-of-the-art approach in this genre is the “Multilingual Universal Sentence Encoder” (MUSE) (Yang et al., 2020). MUSE embeds texts in 16 languages, which are later fed into classification heads for the target task. In particular, the approach is trained for 3 types of tasks: semantic retrieval, bitext retrieval and question answering retrieval. The study also showed that the model can learn cross-lingual vector representations, i.e. it is possible to measure the semantic similarity between two texts of different languages. Despite this cross-lingual resource, we only used the model to embed Portuguese texts about Diabetes Mellitus. In particular, we used the MUSE approach based on the Transformer architecture (Vaswani et al., 2017).

Different from BERT approaches, we followed Yang et al. (2020) and measured the similarity between the vector representations of premise questions and their question-answer candidate pairs using dot product instead of the cosine similarity.

3.3 Learn-to-rank Ensemble (Stacking)
We sought to investigate whether ensembling (i.e., stacking) the semantic similarity measures computed by the previous RQE approaches could boost the results for ranking candidate question-answer pairs to their corresponding premise questions. To fulfill this goal, we trained a Coordinate Ascent...
learn-to-rank approach (Metzler and Croft, 2007). Besides feeding the learn-to-rank approach with semantic features, we also wanted to analyse whether features which assess the quality of the questions can help the RQE task. Relying on Dalip et al. (2012), we used three types of textual quality assessment features as explained next.

3.3.1 Features

Our learn-to-rank approach uses four types of features, one based on the semantic measures computed by the previously described approaches and 3 other which assess the quality of texts based on its length, style and readability. The quality assessment features were computed for both premise questions and their corresponding question-answer candidate pairs.

Semantic Features the semantic similarities computed by our token-based, skip-gram wang2vec, BERT-based, MUSE approaches are used as semantic features by our learn-to-rank ensemble approach.

Length Features we compute several quality assessment features based on length such as the size of a text based on the number of phrases, words and characters.

Style Features we count the total number of phrases higher/lower than the average phrase length in the text; the size of the largest and shorted phrases; the number of articles; prepositions; auxiliary and total number of verbs; coordination, subordinating and correlative conjunctions; indefinite, interrogative, relative and total number of pronouns; an sentences starting with articles, prepositions, auxiliary verbs, general verbs, (coordination, correlative and subordinating) conjunctions and (indefinite, interrogative and relative) pronouns.

Readability Features we computed ARI, Coleman-Liau, Flesch Reading Ease, Flesch Kincaid, Gunning Fog Index, Lasbarhets index and SMOG Grading.

3.3.2 Settings

We trained our Coordinate Ascent learn-to-rank approach with 5 random restarts, 25 iterations to search in each dimension, 0.001 of tolerance and normalizing input features using z-score. The training data for the ensemble of methods was produced with nested cross-validation in the training set, thus avoiding any potential risk of data leakage problems from test to training.

3.4 Fine-tuned Approach

All the embedding approaches described in Section 3.2 and MUSE (Section 3.2.3) were used in a zero-shot learning setup, i.e. they were not trained specifically to our domain and, in the case of the embedding-based approaches, nor even to a semantic retrieval task such as RQE.

In order to have a fine-tuned embedding-based approach in our analysis, we used a neural classifier based on BERTimbau (in its large and cased version) as a Portuguese RQE approach about Diabetes. Given a premise question and a candidate question-answer pair, BERTimbau works by first encoding both documents. Following other fine-tuning studies with BERT (Devlin et al., 2019), for each document, we chose its embedding representation based on its special token [CLS]. Finally, we fine-tune the model by computing the cosine embedding loss function as in Equation 1:

\[
loss(x, y) = \begin{cases} 
1 - \cos(x_1, x_2), & \text{if } y = 1 \\
\max(0, \cos(x_1, x_2)), & \text{if } y = 0 
\end{cases}
\]

where \(x_1\) and \(x_2\) are the [CLS] vector representations of the premise question and the candidate question-answer pair, and \(y\) is the gold-standard label indicating whether they are similar or not. We treat the problem as binary, merging perfect match and relevant cases of our benchmark as positive instances, whereas the irrelevant ones as negatives. During training, the model backpropagates the gradients of the neural network using the AdamW optimizer with learning rate of 1e-5 and batch size of 4.

4 Evaluation

We evaluated the proposed approaches as a ranking problem. Given a premise question from the described benchmark, our goal is to investigate which model can better rank “Perfect Match” and “Relevant” candidate question-answer pairs ahead of “Irrelevant” ones. Following Nakov et al. (2016, 2017), we treated the problem as a binary one, not distinguishing “Perfect Match” and “Relevant” candidate questions.
Table 1: MAP@5 and MRR@5 results of the approaches measuring the similarity among premise questions and candidates through Question-Question, Question-Answer and Question-Question+Answer. Ranking was computed based on pair-wise comparisons among the MAP@5 models with the Wilcoxon Signed-Rank test. Best results are in **bold**, including statistical ties.

## Metrics
We evaluated the approaches using two popular ranking measures: the Mean Averaged-Precision (MAP) as the main metric and the Mean Reciprocal Rank (MRR) as the secondary one. Since each premise question of our benchmark is attached to 5 candidates, we used this length to compute the metrics (e.g., MAP@5 and MRR@5).

## Comparing Strategies
The task of Recognizing Question Entailment (RQE) traditionally works by measuring the similarity between two questions. However, in the Semeval task 3 shared-task about Community Question-Answering (Nakov et al., 2016, 2017), some of the leading approaches worked by measuring the similarity of a premise question taking into account both the candidate question and the answer. Moreover, approaches such as Yang et al. (2020) performs the Question-Answering task as a “Recognizing Answer Entailment” style, where the representation of a premise question is directly compared to the representations of the candidate answers. In this study we investigate the three comparing strategies: Question-Question, Question-Answer and Question-Question+Answer.

## Cross-validation
The approaches were evaluated using cross-validation using 5 folds. The obtained results were averaged across the folds and statistically tested according to the Wilcoxon Signed-Rank test in MAP@5.

## 5 Results
### Overall Analysis: Ranking of Methods
Table 1 displays the MAP@5 and MRR@5 results of the approaches in the Question-Question, Question-Answer and Question-Question+Answer strategies. Best results are marked in **bold**, including statistical ties. Regarding the proposed approaches, results show the advantage of MUSE, being the only method together with the ensemble with semantic features (e.g. L2R Semantic) to rank first in the three strategies according to the MAP@5. Although there is a tie between both methods, MUSE is a single model, whereas the latter is an ensemble of all our semantic similarity, demanding much more computational resources. For this reason, we assume MUSE as the model with the best results in our benchmark. Interestingly, MUSE was applied to the problem of Portuguese QA about Diabetes Mellitus in a zero-shot learning setup, i.e. it was not optimized to the task and, even so, was ranked first in all the strategies.

### Traditional Token-Based Methods
The “old-school” BM25 had very competitive results. The approach ranked first in the Question-Answer and Question-Question+Answer strategies and second in the Question-Question one. Besides effective, this approach has the advantage of not demanding a high volume of computational resources.

### Word Embeddings and BERT
Traditional context-free word embeddings, represented by Wang2vec, did not have a good performance in the evaluation, being outperformed by traditional methods such as BM25 and TF-IDF cosine. Regarding context-sensitive word embedding methods, we evaluated a multilingual version of BERT and two Portuguese focused ones: BERTimbau and BioBERT pt. BERTimbau, a general Brazilian Portuguese-focused model, was the one which performed best among the three, ranking second in the strategies where the candidates were represented by their questions (Question-Question and Question-Question-Answer). BioBERT pt is a Portuguese model pretrained in clinical texts, which we thought would be an advantage of the model.
However, although pretrained on texts in a domain similar to ours, the nature of the texts seems to be different. The clinical texts used to pretrain the method are more technical and focused on healthcare professionals, whereas our corpus is more related to healthcare patients and the way they pose their questions about Diabetes in social media. We also believe that the small datasets’ size in which those methods were originally pre-trained did not benefit the transformer-based approaches, as been reported in the literature (Cunha et al., 2020, 2021).

Fine-tuned Approach Except for the traditional approaches, the embedding-based ones and MUSE were not trained in our benchmark, being evaluated in a learning setting called zero-shot. Another popular learning strategy aims to fine-tune the weights of a pre-trained large neural network, such as BERT, in a downstream task. In order to know how a fine-tuned approach would perform as a Portuguese RQE method about Diabetes, we have developed and tuned the weights of an RQE classifier based on BERTimbau. Together with MUSE and L2R Semantic, this fine-tuned approach (Fine-tuned) ranked first in the Question-Question strategy, outperforming its non-tuned version (e.g., BERTimbau). However, its performance lowered for the other two strategies with results similar to BERTimbau. We believe that the results of our fine-tuned approach was not better in these two strategies, which take candidate answers into account, due to the fact that we trim the input texts with a maximum length of 128 tokens, possibly affecting the representation of the answers in exchange of a faster performance. We leave a deeper analysis of this issue for future work.

Learn-to-rank Ensemble We also sought to investigate whether ensembling (stacking) the semantic similarity measures computed by the proposed RQE approaches could leverage better ranking results. This did not seem to be the case and, in the best situations, the ensemble had comparable results to single approaches such as BM25 and MUSE. Relying on Dalip et al. (2012), we also investigated whether quality assessment features could positively influence the ranking process. In fact, results do not confirm this hypothesis with L2R All, with semantic and quality features underperforming when compared to L2R Semantic, with semantic features only, in the Question-Question and Question-

<table>
<thead>
<tr>
<th>questions</th>
<th>$Q_1$</th>
<th>$Q_2$</th>
<th>$Q_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>answers</td>
<td>10/8</td>
<td>15.5/12</td>
<td>29/19</td>
</tr>
<tr>
<td>answers</td>
<td>59.5/56</td>
<td>86.5/73</td>
<td>102/96</td>
</tr>
</tbody>
</table>

Table 2: Length of questions and answers distributed by quartiles. Each cell contains the pair (length of failure cases/length of success cases).

Question+Answer strategies.

Comparing Strategies Regarding the three compared strategies, results are inconclusive about which one is the best due to a high variation among the approaches. Due to the variability in terms of ranking precision between the three strategies, a choice of strategy could be made based on efficiency. In this case, the Question-Question strategy will be chosen, since, in our benchmark corpus, candidate questions have an average of 22 tokens, being much faster to process than candidate answers, with an average length of 94 tokens.

5.1 Failure Cases Analysis

We also conducted a failure cases analysis when leveraging the best-evaluated method (MUSE). A failure case happens when a non-relevant candidate question or answer is ranked at the top. Once identified, these cases were contrasted with the success cases (when a relevant candidate appears in the first position of the ranking) according to the following criteria: input length distribution, number of relevant candidates and separability.

Input Length Distribution defines the amount of processing required to understand the sentence semantics. Longer sentences require understanding more context and demand more memory to connect different concepts distributed over the sentence. In all strategies mentioned in Section 4, the length of questions and answers of the failure cases are larger than the success cases, as showed in Table 2. In the third quartile, for instance, the questions are 34.48% longer for the failure cases when compared to the success cases.

![Figure 1: Number of relevant candidates per failure/success cases.](image)
Number of Relevant Candidates  the number of relevant candidates also has an important impact on the model’s effectiveness since with less relevant candidates, it is more challenging to place a relevant result in the first ranking position. As demonstrated in Figure 1, the majority of failure cases (75%) had at most 3 relevant candidates with 50% of the samples having at most 1 relevant candidate. On the other hand, in success cases, there were at least 3 relevant candidates for 50% of the samples, with 25% having 5 relevant candidates.

Separability  separability has to do with the ability to generate representations for similar sentences that are closer in the embedding space than semantically dissimilar sentences. We measured the amount of dispersion of similarity scores across all three strategies and the failure cases are up to 25% less separable than success cases. Since with a lower separability is harder to distinguish relevant from non-relevant candidates, the ranking produced by the model diverges from the optimal form.

Summarizing, failure cases seem to be those in which the semantic meaning is more distributed on longer sentences, containing a smaller number of relevant candidates and that are less separable.

6  Related Work

Recognizing Question Entailment (RQE) has been extensively investigated in the field so that it was included as a shared task in SemEval-2016/2017 (Task 3 - Subtask B) (Nakov et al., 2016, 2017). Under the domain of the Qatar Living corpus, the task consisted of reranking 10 candidate question-answer pairs retrieved by Google for a premise question about Qatar. Among the promising participant approaches, we highlight SimBOW (Charlet and Damnati, 2017), the winner of the shared-task in SemEval 2017. The approach works by computing the semantic similarity over the vector representations of a premise question and a corresponding candidate question-answer pair using the SoftCosine metric. Another promising participant was KeLP (Filice et al., 2016), an approach based on Tree Kernels and SVMs which provided top results in the task. After the shared-tasks, Kunneman et al. (2019) conducted a study with these approaches in order to understand the effects of particular design choices, such as the adopted preprocessing methods and word-similarity metrics.

In the medical domain, Wang et al. (2016) proposed an answer recommendation algorithm for medical community question answering. Given a user query, the system starts by looking for similar archived questions using a paragraph vector based language model (PVLM) as a similarity metric. This metric measures the distance between a premise question and a candidate one by multiplying the cosine distances among the word embedding of each word of the premise question with the paragraph vector of the archived question. In the same year, Abacha and Demner-Fushman (2016) proposed a supervised machine learning approach which classifies whether or not a candidate question can be inferred from a premise question. The questions were represented based on lexical and semantic features. More recently, Ben Abacha and Demner-Fushman (2019) proposed a siamese neural network to predict whether a candidate question is a perfect match, a relevant one or an irrelevant one to a premise question.

7  Conclusion

This study investigates Recognizing Question Entailment approaches to build a Community Question Answering about Diabetes Mellitus. Unlike previous studies in the field, ours focuses on a language other than English. Specifically, we focused on the Portuguese language. Due to the lack of resources for the language in this domain, we built a benchmark corpus, which was used to test several RQE models ranging from traditional information retrieval methods to novel large pre-trained language models and ensemble techniques using learn-to-rank techniques. Results show the power of multilingual and multi-task large neural networks such as the MUSE. This sentence encoder obtained the best results of our evaluation in a zero-shot learning setup, i.e. this means it was not optimized to the target task. Results of the evaluation also show that BM25, a traditional and light information retrieval method, can obtain competitive results in the task.

Different from what was expected, state-of-art fine-tuned methods such as our BERTimbau classifier did not perform better than our MUSE zero-shot approach. We believe this may be caused by lack of training data, since our benchmark is relatively small. In future work, we plan to overcome this problem by collecting more data and expanding the corpus to other conditions related to Diabetes, such as hypertension. Like Yang et al. (2020), we also plan to augment our Portuguese training corpus by translating English questions from corpora such as MedQuad into Portuguese.
Acknowledgments

This research was partially funded by the Brazilian agencies CNPq, CAPES, and FAPEMIG. In particular, the researchers were supported by CNPQ grant No. 310630/2017-7, CAPES Post doctoral grant No. 88887.508597/2020-00, and FAPEMIG grant APQ-01.461-14.

References

Asma Ben Abacha, Eugene Agichtein, Yuval Pinter, and Dina Demner-Fushman. 2017. Overview of the medical question answering task at trec 2017 liveqa. In TREC.


On the Usability of Transformers-based models for a French Question-Answering task

Oralie Cattan\textsuperscript{1,2}, Christophe Servan\textsuperscript{1}, Sophie Rosset\textsuperscript{2}

\textsuperscript{1}QW ANT
61 rue de Villiers,
92200 Neuilly-sur-Seine, France
initial.lastname@qwant.com

\textsuperscript{2}Université Paris-Saclay,
CNRS, LISN,
91405, Orsay, France
lastname@lisn.fr

Abstract

For many tasks, state-of-the-art results have been achieved with Transformer-based architectures, resulting in a paradigmatic shift in practices from the use of task-specific architectures to the fine-tuning of pre-trained language models. The ongoing trend consists in training models with an ever-increasing amount of data and parameters, which requires considerable resources. It leads to a strong search to improve resource efficiency based on algorithmic and hardware improvements evaluated only for English. This raises questions about their usability when applied to small-scale learning problems, for which a limited amount of training data is available, especially for under-resourced languages tasks. The lack of appropriately sized corpora is a hindrance to applying data-driven and transfer learning-based approaches with strong instability cases. In this paper, we establish a state-of-the-art of the efforts dedicated to the usability of Transformer-based models and propose to evaluate these improvements on the question-answering performances of French language which have few resources. We address the instability relating to data scarcity by investigating various training strategies with data augmentation, hyperparameters optimization and cross-lingual transfer. We also introduce a new compact model for French FrALBERT which proves to be competitive in low-resource settings.

1 Introduction

Recent advances in the field of Natural Language Processing (NLP) have been made with the development of transfer learning and the availability of pre-trained language models based on Transformer architectures (Vaswani et al., 2017), such as BERT (Devlin et al., 2019). As they provide contextualized semantic representation they contribute both to advance the state-of-the-art on several NLP tasks and also to evolve training practices through the use of fine-tuning.

The trend of recent years consists in training large pre-trained language models on ever larger corpora, with an ever-increasing amount of parameters, which requires considerable computational resources that only a few companies and institutions can afford. For example, the base model of BERT with 110 million parameters was pre-trained on 16 gigabytes (GB) of text, while the GPT-3 model (Brown et al., 2020) was pre-trained on 45 terabytes (TB) of text and has 175 billion parameters.

In fact, deploying ever-larger models raises questions and concerns about the increasing magnitude of the temporal, financial, and environmental cost of training and usability (Strubell et al., 2019; Moosavi et al., 2020). Typically, due to their resource requirements, these models are trained and deployed for industrial operations on remote servers. This leads to a high use of over-the-air communications, which are particularly resource-intensive (Gündüz et al., 2019). In particular, some NLP applications (speech recognition, speech to text, etc.) have some known problems related to network latency, transmission path difficulties, or privacy concerns. To reduce the impact of these communications, there is a solution that is to allow these models to run directly on peripheral or mobile devices, that is, in environments with limited resources that require lightweight, responsive models and energy efficiency. Reducing the size of the models is therefore one of the increasingly favoured avenues, especially for the reduction of memory resources and computation time involved in training and use.

To meet these constraints, compact models represent one of the most promising solutions. As far as we know, they have only been evaluated on the comprehension tasks covered by GLUE (Wang et al., 2018).
et al., 2018) and the question-answering task with the SQuAD corpus (Rajpurkar et al., 2016) with abundant data, in English. The improvements resulting from the algorithmic optimizations of the models, although significant, raise questions about their effectiveness on lower-scale learning problems on poorly endowed languages. The works of Zhang et al. (2021) and Mosbach et al. (2021) have furthermore shown degraded performance in these conditions. These two reflections are at the origin of a double question which our contribution attempts to answer. On the one hand, what is the behavior of a Transformer-based model in the context of a question-answering task in French, a task that is poorly endowed in this language? On the other hand, what are the impacts of algorithmic improvements of these same models in this context?

To answer these questions, we first establish in section 2 a state-of-the-art that is meant to be broad enough to have a shallow overview depicting the ins and outs and issues around the usability of Transformer-based models whose breadcrumb trail is the issue of resources. Then, we present in the section 3 the recent progress of the question-answering task, through the use of these latest models. In sections 4 and 5 we introduce our model and present our experiments on the usability of Transformers models in a question-answering task for French, on FQuAD (d’Hoffschmidt et al., 2020) and PIAF (Keraron et al., 2020) corpora. We propose to address the instability relating to data scarcity by investigating various training strategies with data augmentation, hyperparameters optimization and cross-lingual transfer. Finally, we present a new compact model for French based on ALBERT (Lan et al., 2020)\(^1\), and compare it to existing monolingual and multilingual models, large and compact, under constrained conditions (notably on learning data).

2 Usability of Transformers

In this section we present the ins and out of the Transformer models to understand how the approaches meet the need for better usability.

2.1 Architecture and pre-trained models

The Transformer architecture (Vaswani et al., 2017) is based on a stack of encoder-decoder blocks, composed at a high level of forward propagation networks and multi-headed self-attention operations. The self-attention layer is the core element of its architecture that enables its efficiency in modeling the semantic context interdependencies between the units or sub-units of the input sequence.

Transformer-based language models such as BERT (Devlin et al., 2019) are pre-trained on large-scale data collections sourced from Wikipedia or Common Crawl (CC) with one or multiple training objectives (masked language modeling, next sentence or sentence order prediction). This pre-training can be followed by supervised fine-tuning according to the tasks, whether generatives (machine translation, abstractive summarization) or discriminatives (classification, question-answering). The ensuing fine-tuning phase allows for better initialization of the models parameters while requiring less task-specific data so as to make the training of subsequent tasks faster.

Recently, Zhang et al. (2021) and Mosbach et al. (2021) have nevertheless shown that the commonly adopted practices (the number of iterations, the choice of model layers) when fine-tuning Transformers-based language models are inappropriate under resource constrained conditions and adversely affect the stability of models performances as overfitting, label noise memorization or catastrophic forgetting. Added to this, because the pre-training process is particularly constraining, various works have been oriented towards the research and training of efficient models, both in terms of available capacities and resources and in terms of environmental footprint.

2.2 A search for efficiency

Reducing the cost of training Transformers-based models has become an active research area. To this end, methods based on compression techniques or on architecture improvements have been introduced in order to build compact models with comparable performances to large models.

Many works address the issue of model compression with quantization, pruning, knowledge distillation or a combination of these approaches. The idea of quantization (Shen et al., 2020) is to take advantage of the use of lower precision bit-width floats to reduce memory usage and increase computational density. Following the same objective, pruning (Michel et al., 2019) consists in removing parts of a model (weight bindings, attentional heads) with minimal precision losses. Finally, knowledge distillation (Sanh et al., 2019) enables the generation

\(^1\)Available at HuggingFace’s model hub page.
of models that mimic the performance of a large model (or set of models) while having fewer parameters.

Another axis of development concerns the use of neural architecture search (Elsken et al., 2019) which allows to optimize a model by progressively modifying the design of the network through trial and error, eliminating insignificant operations. To avoid the unnecessary large number of parameters, adapters (Houlsby et al., 2019) were introduced to allow fine-tuning of the set of parameters specific to the task of interest rather than the entire model.

Other architectural improvements highlighted with the introduction of the ALBERT model (Lan et al., 2020) such as the factorization of the attention matrix or parameter sharing. Indeed, the most time-consuming and memory-intensive operations concerns the forward propagation and attention computation operations. The self-attention layer of BERT pretrained models grows quadratically in respect to the input sequence length. One common approach to this issue consists of approximating the dot-product attention for example by using hashing techniques (Kitaev et al., 2020) to accelerate the training and inference phases when long sequence lengths are used. However these solutions have demonstrated they suffer from important computational overheads for tasks with smaller lengths, such as question-answering.

3 The Question-Answering task

Question-Answering (QA) based on machine reading comprehension corresponds to the task of extracting an answer given a question and a context document such as from a news or Wikipedia article.

3.1 General QA Architecture

Until recently, most of the proposed approaches have relied on an architectural complexification of LSTM-based neural networks and attention mechanism. At a high level, their architectures are all composed of three layers with:

(a) an encoding layer that projects the inputs, as each word within the context, question and answer triples in a latent semantic space;

(b) an interaction layer that models the semantic interdependencies between the embedded inputs through the use of attention mechanisms.

(c) an output layer that extracts the answer to the input question within the related context.

The interaction layer is the core element of the architecture for which several kinds attention mechanisms has been developed to improve the QA matching process such as bi-attention (Seo et al., 2017), co-attention (Xiong et al., 2017, 2018), multi-level inter-attention (Huang et al., 2018) or re-attention (Hu et al., 2018), to name just a few.

Recent advances through the availability of Transformer-based pre-trained models and the development of transfer learning methods have enabled to remove the recurrence of previous architectures in order to achieve parallelization efficiencies. This simplified the QA architecture and its training process, replacing the encoding and the interaction layers with attention-based Transformer layers.

Another advantage is that it provides pre-computed contextual word representations. QA models based on LSTMs are built on top of static word embeddings models such as GloVe (Pennington et al., 2014). Even these models have up to 40 times fewer parameters than a BERT-based model, they rely on LSTM-based encoders to produce contextual embeddings which considerably lengthens the time required for training and makes the dependence on supervised data more important.

The standard approach introduced by Devlin et al. (2019) we rely on this study, consists in introducing and updating parameter vectors corresponding to the start and end positions of the answer span. Specifically, the start and end position probability distributions are computed by softmax over the dot products between the representation of the tokens and the start and end vectors. In sum, all of the Transformer parameters as well as the two introduced parameter vectors are optimized together.

Despite the fact that there is a number of large-scale QA datasets in English, with tens of thousands of annotated training examples, porting a system to a new language with fewer annotated resources (low-resource languages) requires approaches that go far beyond the simple act of retraining the models.

3.2 Low-Resourced QA

In recent years, low-resource NLP has drawn an increasing amount of attention with solutions ranging from developing new data collection methodologies either via crowdsourcing or through the use of machine translation (MT), to cross-lingual and transfer learning approaches for which information is shared across languages or tasks.
3.2.1 MT-based data collection

Neural MT as made considerable progress in recent years such as translating large-scale datasets from a high-resourced to under-resourced languages or conversely has become an intuitive way of generating annotated datasets in a cost-effective and rapid manner.

Automatically translating the context, question and answer triples from a high-resource language, such as English (called source domain) to low-resource languages (called target domains) have enabled the evaluation of models for languages with no training data available but also the creation of large-scale MT-based QA corpora for the Italian (Croce et al., 2018), Spanish (Carrino et al., 2020), Arabic (Mozannar et al., 2019) and Korean (Youngmin Kim, 2020) languages.

Another approach consists of translating the QA triples of the target domain into the source domain, so the model trained on the source language can be directly applied on the translated target language testing data. As an example, Asai et al. (2018)’s method consisted of combining the alignment attention scores from a MT model with an English QA model to guide the answer extraction process.

The performance of MT-data approaches depends strongly on the quality of the MT models. Thus, due to the lack of reliable models for some language pairs, approaches that foster the transfer of knowledge from other languages or tasks while requiring less data have been developed.

3.2.2 Pre-training and Transfer approaches

The exploitation of pre-trained models followed by task-specific fine-tuning have pushed the state-of-the-art forwards, while requiring much less computational and data resources. The idea behind pre-training is to reuse the weights parameters trained on a set of source tasks and continue to fine-tune them on under-resourced target tasks to achieve knowledge transfer. Dai and Le (2015) were the first to propose to pre-train RNNs using auto-encoders and language models as part of their QA encoding layer. Min et al. (2017) and Wiese et al. (2017) pre-trained QA models before applying the fine-tuning process between the source and the target domains. Other efforts focused on pre-training Transformer-based models multilingually such as the multilingual version of BERT (called mBERT) (Devlin et al., 2019) or XLM-R (Conneau et al., 2020) to learn cross-lingual representations which are transferable across languages.

3.2.3 Usability concerns

Studies on the usability of Transformer-based models (Section 2) from standard resource efficiency concerns towards a broader set of problems related to their generalizability.

Recently, Pires et al. (2019) and Conneau et al. (2020) have shown that multilingual models underperformed, when applied on poorly endowed languages. Additionally, as mentioned in subsection 2.1, recent works (Zhang et al., 2021; Mosbach et al., 2021) have highlighted the limitation of Transformer-based transfer learning with strong instabilities arising from the small-scale learning.

French is a poorly endowed language since we do not have enough annotated data to train a deep learning model on QA tasks. Moreover, unlike the only two large monolingual French models: CamemBERT (Martin et al., 2020) and FlauBERT (Le et al., 2020), the English BERT model has become a branching point from which a growing number of large and compact English pre-trained models have emerged. These French monolingual models, although they provide good performances, do not reflect the rapid evolution of the field.

Consequently, in this paper we propose a new compact model FrALBERT for French we present in the following section alongside with the other available pre-trained language models on which we base our experiments.

4 FrALBERT and Transformer models considered

As mentioned in the previous section there is no compact model for French. We therefore decided to pre-train a new version of ALBERT from scratch we called FrALBERT, thus overcoming some of the discussed limitations.

ALBERT is based on parameter sharing/reduction techniques that allows to reduce the computational complexity and speed up training and inference phases. Compared to previous compact models such as DistilBERT (Sanh et al., 2019), Q-BERT (Shen et al., 2020) or TernaryBERT (Zhang et al., 2020), ALBERT is to the date the smallest pre-trained models with 12 million parameters and <50 megabyte (MB) model size.

FrALBERT is pre-trained on the French version of the Wikipedia encyclopedia of 04/05/2021, i.e. 4 GB of text and 17 million (M) sentences. Beyond concerns about the rights to use data from Common
<table>
<thead>
<tr>
<th>model</th>
<th>pre-training data</th>
<th>vocab. size</th>
<th># param.</th>
<th>model size</th>
</tr>
</thead>
<tbody>
<tr>
<td>CamemBERT&lt;sub&gt;base&lt;/sub&gt;</td>
<td>French OSCAR (138 GB of text)</td>
<td>32005</td>
<td>110 M</td>
<td>445 MB</td>
</tr>
<tr>
<td>CamemBERT&lt;sub&gt;large&lt;/sub&gt;</td>
<td>French CCNet (135 GB of text)</td>
<td>32005</td>
<td>335 M</td>
<td>1.35 GB</td>
</tr>
<tr>
<td>CamemBERT&lt;sub&gt;base&lt;/sub&gt;</td>
<td>French Wikipedia (4 GB of text)</td>
<td>32005</td>
<td>110 M</td>
<td>445 MB</td>
</tr>
<tr>
<td>FrALBERT&lt;sub&gt;base&lt;/sub&gt;</td>
<td>French Wikipedia (4 GB of text)</td>
<td>32005</td>
<td>12 M</td>
<td>50 MB</td>
</tr>
<tr>
<td>XLM-R&lt;sub&gt;base&lt;/sub&gt;</td>
<td>CC-100 (2.5 TB of text)</td>
<td>250002</td>
<td>278 M</td>
<td>1.12 GB</td>
</tr>
<tr>
<td>XLM-R&lt;sub&gt;large&lt;/sub&gt;</td>
<td>CC-100 (2.5 TB of text)</td>
<td>250002</td>
<td>559 M</td>
<td>1.24 GB</td>
</tr>
<tr>
<td>mBERT&lt;sub&gt;base&lt;/sub&gt;</td>
<td>Wiki-100</td>
<td>119547</td>
<td>177 M</td>
<td>714 MB</td>
</tr>
<tr>
<td>small-mBERT&lt;sub&gt;base&lt;/sub&gt;</td>
<td>Wiki-100</td>
<td>33407</td>
<td>111 M</td>
<td>447 MB</td>
</tr>
<tr>
<td>distil-mBERT&lt;sub&gt;base&lt;/sub&gt;</td>
<td>Wiki-100</td>
<td>119547</td>
<td>134 M</td>
<td>542 MB</td>
</tr>
</tbody>
</table>

Table 1: Characteristics of the pre-trained models used in the proposed small-scale QA framework.

Our experiments are also based on the large monolingual French model CamemBERT (Martin et al., 2020) as well as on the two large multilingual models: XLM-R (Conneau et al., 2020) and mBERT (Devlin et al., 2019), both pre-trained from massive corpora dataset in more than 100 languages such as the Common Crawl (CC-100) or Wikipedia (Wiki-100). We also exploit two compact multilingual models with a distilled version of mBERT: distil-mBERT (Sanh et al., 2019) and small-mBERT (Abdaoui et al., 2020), a mBERT model whose the original vocabulary has been reduced to two languages (English and French). Table 1 gives a comparison of the models.

5 Experiments

We propose an assessment of the comparative advantage gains in performance when using different training strategies (data augmentation, hyperparameter search and cross-lingual transfer) over monolingual and multilingual pre-trained models, larges and compacts for a QA task in French under resource constraints.

5.1 QA Datasets

We conduct experiments on four QA datasets whose descriptives are presented in Table 2, with:

- **SQuAD** (v1.1) (Rajpurkar et al., 2016) (we called *SQuAD-en*) the reference corpus to evaluate QA models’ performances in English, consisting of 100K+ QA pairs sourced from 442 English Wikipedia articles;

- **FQuAD** (v1.0) (d’Hoffschmidt et al., 2020), a recently released French QA dataset consisting of 25K+ crowdsourced QA pairs based on 135 articles on French Wikipedia;

- **PIAF** (v1.0) (Keraron et al., 2020), a small-scale dataset in French with only 3K+ pairs of QA pairs in 191 Wikipedia articles; and

- **SQuAD-fr<sub>train</sub>**, our French translated version of *SQuAD-en*. We used the Transformer architecture as described in Vaswani et al. (2017) from the Open NMT framework (Klein et al., 2017) (Open-Source Neural Machine Translation) implementation of the network to train our neural MT system. When translating QA corpora, the problem we face is that the translated answer may not be present in the translated context. Thus, simple techniques such as segment matching are inadequate to retrieve the answer. We have developed an answer extraction process that is based on

<table>
<thead>
<tr>
<th>dataset</th>
<th><em>SQuAD-en</em>&lt;sub&gt;train&lt;/sub&gt;</th>
<th>FQuAD&lt;sub&gt;train&lt;/sub&gt;</th>
<th>FQuAD&lt;sub&gt;dev&lt;/sub&gt;</th>
<th>PIAF&lt;sub&gt;dev&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td># titles</td>
<td>442</td>
<td>117</td>
<td>18</td>
<td>191</td>
</tr>
<tr>
<td># paragraphs</td>
<td>18,891</td>
<td>4,920</td>
<td>768</td>
<td>761</td>
</tr>
<tr>
<td># sentences / # tokens / #characters</td>
<td>5.0 / 119.7 / 635.6</td>
<td>4.8 / 125.9 / 653.8</td>
<td>5.4 / 147.3 / 765.7</td>
<td>4.2 / 110.5 / 579.1</td>
</tr>
<tr>
<td># questions</td>
<td>87,342</td>
<td>20,088</td>
<td>3,184</td>
<td>3,812</td>
</tr>
<tr>
<td># tokens / # #characters</td>
<td>10.1 / 50.7</td>
<td>9.2 / 47.9</td>
<td>8.5 / 45.6</td>
<td>9.1 / 48.2</td>
</tr>
<tr>
<td># answers</td>
<td>87,599</td>
<td>20,731</td>
<td>3,188</td>
<td>3,835</td>
</tr>
<tr>
<td>human performance (F1 / EM)</td>
<td>90.5 / 80.3</td>
<td>92.6 / 79.5</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2: Descriptives of *SQuAD-en*, FQuAD and PIAF datasets.
ChrF (Popović, 2015) a character n-gram precision and recall enhanced with word n-grams. Since answers are largely made up of entities, ChrF score integration is only performed when the answer span is not present in the related context. In order to evaluate the quality of the translation, we manually corrected the translation errors in the output of a subset of the corpus composed of 890 QA pairs and 107 contexts. We obtain a BLEU score (Papineni et al., 2002) of 68.89 and 72.38 for questions and contexts respectively. SQuAD-fr \(F_{\text{train}}\) serves as a means of data augmentation on FQuAD and PIAF benchmarks, with 90K+ translated QA training pairs.  

We also explore mixed datasets training strategy with SQuAD-en \(F_{\text{train}}\) for training models on a concatenation of the training data covering French-English language pairs to test the cross-lingual transfer ability of multilingual models.

### 5.2 Evaluation and validation

The performance of QA models are evaluated using the Exact Match (EM) and F1 scores. The EM score is the percentage of system outputs that match exactly with the ground truth answers. The F1 score is a combined measure of precision and recall that is less strict than EM. The evaluation process involves post-processing identical to that presented by d’Hoffschmidt et al. (2020) and inspired by that proposed for English by Rajpurkar et al. (2016), which consists of the removal of punctuation marks and determiners as well as a down-casing of the answers (ground truths and predictions).  

To address our considerations related to resource constraints we perform a hyperparameter optimization, that has proven to lead to better solutions in less time. It is based on a population-based learning (Jaderberg et al., 2017) in which a population of models and their hyperparameters are jointly optimized. To this end, we build a validation set by randomly extracting 10% of the training data.

---

2We share our SQuAD-fr corpus on request and on dataset sharing platforms to support further research in this area.

3The experiments reported in (d’Hoffschmidt et al., 2020) concern version 1.1 of the FQuAD corpus. Ours are based on the only version available to date (v1.0). Moreover, the test sets are not made public, so we use the development set instead.

4Determiners are le, la, les, l’, du, des, au, aux, un, une.

### 5.3 Results

Table 3 presents the results on the French QA task evaluated on FQuAD \(F_{\text{dev}}\) and PIAF \(F_{\text{dev}}\). This table shows the scores obtained with Transformer-based models on the baseline training (FQuAD \(F_{\text{train}}\)), using hyperparameter optimization approach (FQuAD \(F_{\text{train}}\) w/ opt) and with data augmentation approach (FQuAD \(F_{\text{train}}\) + SQuAD-fr \(F_{\text{train}}\)). Cross-lingual performances are presented in table 4, on the same French QA tasks FQuAD and PIAF as baseline the English corpus SQuAD-en \(F_{\text{train}}\), on which we applied hyperparameter optimization (SQuAD-en \(F_{\text{train}}\) w/ opt) and performed data augmentation by adding the French corpus FQuAD to the English QA training corpus (SQuAD-en \(F_{\text{train}}\) + FQuAD \(F_{\text{train}}\)).

#### 5.3.1 Baseline results

The results obtained from monolingual models with CamemBERT \(\text{base} \) which has more layers, hidden units and attention heads and Camembert \(\text{base} \), both pre-trained with a larger and more diverse amount of data achieve results are better than Camembert \(\text{base} \) pre-trained on only 4 GB Wikipedia. The highest F1 score is 81.2 on FQuAD \(F_{\text{dev}}\) and 68.1 on PIAF \(F_{\text{dev}}\).  

The F1 performances of the FrALBERT \(\text{base} \) model are close to those of the CamemBERT \(\text{base} \) model, both pre-trained on the French content of Wikipedia (4GB). Their results turn out to be competitive and of the same order of magnitude as those reported by Lan et al. (2020) on SQuAD-en with 1 point difference on F1 scores when evaluating a BERT \(\text{base} \) model (90.4 F1) and a compact ALBERT \(\text{base} \) model (89.3 F1) pre-trained on the same texts (BookCorpus and Wikipedia). Interestingly, these EM scores are higher than those of the Camembert \(\text{base} \) achieving the EM score of 55.1, an increase of 5 points on the FQuAD \(F_{\text{dev}}\).

#### 5.3.2 hyperparameter results

Automatically tuning the hyperparameter tends to make QA models more accurate with gains in terms of EM scores that are very expressive. Highest F1 / EM scores are 90.2 / 75.5 on FQuAD \(F_{\text{dev}}\) and 71.0 / 44.8 on PIAF \(F_{\text{dev}}\). Improvement are variable accruing the model considered, especially the French BERT one which have the highest improvement using this approach (from 6 to 9 F1 points and from 11 to 20 EM points) which is quite impressive. FrALBERT stay behind of 5 F1 points of the Camembert \(\text{base} \) trained with the same data (wiki
4 GB) but regarding the EM scores, FrALBERT is better of 3 points. Surprisingly, multilingual models are close to the French BERT models. The small-mBERT is better than FrALBERT around 2.5 F1 points, while the French one have a better EM score (+2.6 points), while distil-mBERT is lower in both F1 and EM scores.

### 5.3.3 Data augmentation results

Training strategies based on data augmentation got nearly the best results in both F1 and EM scores except for the CamemBERT large. Apart from CamemBERTbase (wiki 4 GB) and FrALBERTbase models, results are comparable with an average difference of 1 point regardless of the metric. More generally, the performance gains are up to 11 and 20 of F1 and EM points, respectively, on FQuADdev and up to 4 points on both metrics on PIAFdev.

### 5.3.4 Cross-lingual transfer results

The cross-lingual transfer-based approaches using multilingual models outperform the monolingual approaches on FQuAD and PIAF corpora (table 4). Once again the large model XLM-R achieves better results than its base version. XLM-R pre-trained with a dual LM objective lens scores better than the mBERT model every time. The highest F1 score is 86.8 on FQuADdev and 70.4 on PIAFdev.

There is a significant performance drop between the large multilingual models and their respective compact models. The compact multilingual models based on mBERT substantially underperform, obtaining lower F1 and EM scores than the large models regardless of the training strategy. In the zero-shot configurations where no French data is used for training (SQuAD-en_train and SQuAD-en_train + FQuADtrain) using F1-measure (F1) and Exact Match (EM), on two French QA tasks (FQuADdev and PIAFdev).

### 5.3.5 General observations

In all configurations, the performance in terms of EM and F1 on PIAF remains significantly lower than that obtained on FQuAD since the PIAF corpus does not include multiple responses as pointed out by d’Hoffschmidt et al. (2020). Unsurprisingly, PIAFdev offer a more challenging evaluation set, where the answer extraction performance are lower. Indeed, the corpus is more diversified with questions on 191 different Wikipedia articles, whereas on FQuADdev it only covers 18.

According results, we can confirm that data aug-

---

**Table 3:** Results obtained with French and multilingual Transformer models on the baseline training (FQuADtrain), using hyperparameter optimization (FQuADtrain w/ optim) and with data augmentation (FQuADtrain + SQuAD-frtrain) using F1-measure (F1) and Exact Match (EM), on two French QA tasks (FQuADdev and PIAFdev).

**Table 4:** Cross-language transfer results obtained with multilingual Transformer models only on the baseline (SQuAD-en_train), using hyperparameter optimization (SQuAD-en_train w/ optim) and with data augmentation (SQuAD-en_train + FQuADtrain) using F1-measure (F1) and Exact Match (EM), on two French QA tasks (FQuADdev and PIAFdev).
mentation is the better way to improve results, even if data comes from another language. We observe from multiligual results that combining training data gives better results with similar performance whether data are translated or not.

5.4 Analysis
In this section, we conduct an analysis of our results to understand what remains as challenges for state-of-the-art models, with a focus on the usability concerns of Transformers under resource constraints.

The success of supervised methods depends heavily on the availability of large-scale training data. Pre-training large models on massive corpora using unsupervised language modeling and fine-tuning the model with pre-trained weights requires less task-specific data. Our experimental results are in line with this, since we obtain satisfactory results with transfer learning when large and high quality annotated data are not available. The highest F1 score is 81.2 on FQuAD\textsubscript{dev} and 68.1 on PIAF\textsubscript{dev}. Nevertheless, the performance of the models can benefit from several training strategies.

Effect of MT-data augmentation The lack of human-annotated datasets for languages other than English can be overcome by enriching our training data with the translated version of SQuAD\textsubscript{-en}\textsubscript{train}. Regardless of the pre-trained model used, their performance are competitive on FQuAD\textsubscript{dev} and PIAF\textsubscript{dev}, close to human performance.

Effect of hyperparameter tuning A generally unstated assumption is that pre-trained linguistic models are under-optimized and that practices commonly adopted for the fine-tuning stage can be detrimental to performance (Zhang et al., 2021; Mosbach et al., 2021). This is quite apparent in all settings, with better gains through hyperparameter optimization stages. Fine-tuning CamemBERT\textsubscript{large} on the French dataset yields 90.2 / 75.5 F1 / EM on the FQuAD dev set. By means of comparison, CamemBERT\textsubscript{large} scores were 81.2 / 55.9 F1 / EM on the same set with no hyperparameter tuning.

Crosslingual QA Pre-training language models on the concatenation of multiple languages has proven to be a competitive approach for cross-lingual language modeling. If monolingual models often perform better than multilingual models, we observe that, for comparable model sizes this is not the case in our task where the performance gap is smaller. This gap is further reduced when the strategies are combined. Scores of fine tuned XLM-R\textsubscript{base} is 82.1 / 66.8 on FQuAD\textsubscript{dev} and 65.0 / 39.6 on PIAF\textsubscript{dev}.

The zero-shot experiments show that multilingual models can reach strong performances on the task in French when the model has not encountered data of the French language. For example, the XLM-R\textsubscript{base} model fine-tuned solely on SQuAD\textsubscript{-en}\textsubscript{train} reaches a performance on FQuAD just a few points below the performance obtained when fine-tuning is performed on FQuAD\textsubscript{train}.

Finally, our results suggest that data-driven augmentation, either by translating datasets from high resource languages or by concatenating the available corpora are a particularly appropriate strategy to exploit the potential of cross-lingual transferability of models and data for improving model performances.

Improvements over a small scale dataset With resource-limited training data we obtain an average F1 score of 74.2 on FQuAD\textsubscript{dev} and 61.7 on PIAF\textsubscript{dev} when we fine-tuned FrALBERT — higher performance than any of the compact multilingual models, but slightly below the performance of the large monolingual models. We believe that the lower performance of small multilingual models is not due to their lower number of parameters, but to the usability of these models which is dependent on the reduction technique used. This can be seen very clearly since the performance of the FrALBERT model is close to that of the large models.

Here again, these performances can be boosted via the use of translated data or hyperparameter search which allows us to bring the maximum performances obtained with FrALBERT\textsubscript{base} and CamemBERT\textsubscript{base} pre-trained on 4 GB Wikipedia closer in a consistent way. Their scores remain slightly below those obtained with models pre-trained on more data suggesting limitations related to the corpus domain of the language model.

Computational costs Compact models provide alternatives to high-energy consumption models by showing comparable performance while reducing their size and computational complexity. Decreasing the environmental impact of NLP model training, as a research topic, is very recent (Moosavi et al., 2020). We decided to monitor our experiments conducted on one NVIDIA V100 GPU with 16GB of memory using
Table 5: Comparison of models by computational costs on FQuAD_{train}.

<table>
<thead>
<tr>
<th>model</th>
<th># param.</th>
<th>model size</th>
<th>Time (s)</th>
<th>Energy (kWh)</th>
<th>CO₂ (g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CamemBERT_{base}</td>
<td>110 M</td>
<td>445 MB</td>
<td>7,207</td>
<td>1.08</td>
<td>317.87</td>
</tr>
<tr>
<td>CamemBERT_{large}</td>
<td>335 M</td>
<td>1.35 GB</td>
<td>19,445</td>
<td>3.10</td>
<td>914.27</td>
</tr>
<tr>
<td>FrALBERT_{base}</td>
<td>12 M</td>
<td>50 MB</td>
<td>3,816</td>
<td>0.57</td>
<td>167.80</td>
</tr>
<tr>
<td>XLM-R_{base}</td>
<td>278 M</td>
<td>1.12 GB</td>
<td>7,676</td>
<td>1.14</td>
<td>337.70</td>
</tr>
<tr>
<td>XLM-R_{large}</td>
<td>559 M</td>
<td>2.1 GB</td>
<td>21,137</td>
<td>3.30</td>
<td>973.29</td>
</tr>
<tr>
<td>mBERT_{base}</td>
<td>177 M</td>
<td>714 MB</td>
<td>7,333</td>
<td>1.07</td>
<td>317.02</td>
</tr>
<tr>
<td>small-mBERT_{base}</td>
<td>111 M</td>
<td>447 MB</td>
<td>7,190</td>
<td>1.09</td>
<td>321.42</td>
</tr>
<tr>
<td>distil-mBERT_{base}</td>
<td>134 M</td>
<td>542 MB</td>
<td>6,466</td>
<td>1.06</td>
<td>314.17</td>
</tr>
</tbody>
</table>

The footprint of the large versions of XLM-R_{large} and CamemBERT_{large} models is 3 times more than their base versions. Their training time is also significantly longer, over 5 hours. Finally, in terms of watt usage, carbon emissions and training time, FrALBERT is two times less the distilled version of BERT.

6 Conclusion and outlook

Recently, important progress has been made in neural language modeling using Transformer networks. Its popularity now well established lies in its effectiveness in modeling long-term dependencies. In this study, we have shown that a number of significant shortcomings of usability have recently been pointed out and that some solutions have been drawn up with compact models. We have also overviewed how the use of Transformer-based pre-trained language models have sparked a paradigmatic shift in question-answering training practices from task-specific architectures to the use of transfer learning through fine-tuning.

Comparing performances on a French question-answering task using large and compact models provides insight into the usability of these models for under-resourced languages. As others, we argued that large and compact models cannot be used with limited data. Our experimental results suggest that training strategy such as hyperparameter tuning or data augmentation can help to alleviate the data-gathering burden, with performances close to those of a high-resourced language such as English.

Finally, we present a new compact model for French FrALBERT (12M parameters), which proves to be as competitive as the large monolingual model CamemBERT (110M parameters) pre-trained on the same amount of text. In term of computational cost, we shown this compact model is twice less greedy than the BERT_{base} models. We also release a high-quality translated version of the SQuAD corpus in French consisting of around 90K+ QA pairs.

In a future work, we aim to continue this study from a meta-learning perspective with a model-agnostic approach generalizable to low-resource languages. We also plan to extend our model to other languages and to evaluate it on other NLP tasks such as named entity recognition or natural language understanding.

Acknowledgments

This work has been partially funded by the French National Research Agency (ANR) through the project TextToKids (AAPG2019).

References


258


Classification of Code-Mixed Text Using Capsule Networks

Shanaka Chaturanga, Surangika Ranathunga
Department of Computer Science and Engineering,
University of Moratuwa,
Katubedda 10400, Sri Lanka.
[shanaka.chathuranga.19, surangika]@cse.mrt.ac.lk

Abstract

A major challenge in analysing social media data belonging to languages that use non-English script is its code-mixed nature. Recent research has presented state-of-the-art contextual embedding models (both monolingual s.a. BERT and multilingual s.a. XLM-R) as a promising approach. In this paper, we show that the performance of such embedding models depends on multiple factors, such as the level of code-mixing in the dataset, and the size of the training dataset. We empirically show that a newly introduced Capsule+biGRU classifier could outperform a classifier built on the English-BERT as well as XLM-R just with a training dataset of about 6500 samples for the Sinhala-English code-mixed data.

1 Introduction

Social media has become very popular among people across the world during the last decade, mainly due to the popularity of smart mobile phones. For low-resource languages, this social media data has become a major source of text in building Natural Language Processing (NLP) applications. Most of the content in social media tends to be informal. When the users are (at least to a certain degree) multilingual, the informal content they publish in social media tends to be code-mixed. Code-mixed data is a result of code-switching, which denotes a shift from one language to another within a single utterance (Sitaram et al., 2019).

Recent work on text classification with code-mixed data has used contextual embedding models such as BERT (Devlin et al., 2019a), and their multilingual versions, such as mBERT or XLM-R (Aguilar et al., 2020; Kumar et al., 2020). However, using these pre-trained models for code-mixed text classification, in particular the domain-specific data in low-resource languages poses many challenges. Contextual embedding models such as BERT need large volumes of monolingual data to train. On the other hand, not every language is included in the pre-trained multilingual models, and low-resource languages are underrepresented in those models (due to the smaller amounts of low-resource language data used in contrast to high-resource languages when training these models). In fact, there is a line of research that has shown even simple classifiers such as Logistic Regression proving to be more effective than the multilingual embedding models (Chakravarthi et al., 2020).

In this paper, we empirically show that the success of contextual embedding models on code-mixed text classification depends on multiple factors, and they can indeed be sub-optimal compared to text classification based on neural models other than transformer based ones.

We selected Sinhala and Malayalam, which are low-resource languages. In the recent language categorization by Joshi et al. (2020), Sinhala belongs to class 0 (i.e. it has exceptionally limited resources). Malayalam is categorised as 1, (i.e. it has some unlabelled data, however collecting labelled data is challenging).

A food recipe dataset with about 3000 samples (Kazhuparambil and Kaushik, 2020) was used as the Malayalam-English code-mixed data. For Sinhala-English code-mixed data, a corpus of 10000 user comments in the domain of telecommunication was annotated with two types of information: aspects related to the telecommunication domain, and overall sentiment of the comment.

We fine-tuned English-BERT and XLM-R (Conneau et al., 2019) models on both datasets. Then we implemented a novel Capsule+biGRU network for the same tasks. Results show that the Capsule+biGRU model consistently outperforms English-BERT and XLM-R models for the Sinhala-English dataset that had more data and less code-mixing complexity than the Malayalam-English dataset. With this, we establish the argument that the performance of contextual embedding mod-
els depends on multiple factors such as the code-mixing level, size of the dataset used to train the contextual embedding models, and the size of the dataset used in fine-tuning. Further experiments with the Sinhala-English dataset showed that this Capsule+biGRU model is superior to recurrent models as well. The annotated dataset, as well as our code are publicly released 1

2 Related Work

2.1 Deep Learning based Text Classification

Aspect identification and sentiment classification tasks used in this paper are essentially text classification problems. Thus, without any loss of generality, in this section we look at Deep Learning solutions applied for text classification. Minaee et al. (2021) identified several Deep Learning techniques for supervised text classification. The simplest technique is the Feed Forward networks, which treats the input text as a bag of words. Subsequently introduced Recurrent Neural Models have the ability to capture the sequential dependencies between words. Long-Short Term Memory Networks (LSTMs), and Gated Recurrent Units (GRUs) were introduced to solve some of the shortcomings of the RNN models. LSTMs, in particular bi-LSTMs have been very commonly used. There have been several LSTM variants employed in text classification such as tree-LSTM (Tai et al., 2015), multi-timescale LSTM (Liu et al., 2015), and sentence-state LSTM (Zhang et al., 2018). It is common to use pre-trained word embedding models such as Word2Vec or fastText to be used as the input representation of these recurrent models. Attention mechanism is employed on top of architectures such as GRUs (Yang et al., 2016), or LSTMs (Liu et al., 2016). Convolutional Neural networks (CNNs) is another model used for text classification. Similar to LSTMs, different CNN variants such as character-level CNNs (Zhang et al., 2015), and multi-layer CNNs (Pang et al., 2016) have been employed. However, CNN has a problem of information loss with respect to its pooling operation. More recently, capsule networks were introduced to address this problem, and have been reported to outperform CNN based text classification systems (Yang et al., 2019), as well as those based on recurrent models like LSTMs (Senevirathne et al., 2020). Recently introduced Transformers (Vaswani et al., 2017) are now being commonly used to build extremely large language models (also known as contextual embedding models). The most popular contextual embedding model is BERT (Devlin et al., 2019a), and there have been subsequent improvements to it. Text classification with pre-trained language models has now become the state-of-the-art for Text Classification (Bao et al., 2020).

2.2 Classifying Code-Mixed Data

Research on code-mixed data spans across tasks such as language identification (Gundapu and Mamidi, 2018), Part of Speech tagging (Vyas et al., 2014), speech recognition (Shah et al., 2020) and text classification. In this discussion, we only focus on text classification of code-mixed data. Out of the aforementioned Deep Learning techniques, code-mixed data classification has been mainly implemented using LSTMs, and contextual embedding models. English-BERT was employed because in code-mixed data, foreign language text is written in English script. Moreover, multilingual contextual embedding models s.a. LASER (Artetxe and Schwenk, 2019), mBERT (Devlin et al., 2019b) and XLM-R have been employed. Related research reported mixed observations on the performance of these techniques. Aguilar et al. (2020) showed that mBERT performs better than biLSTM that has one-hot vector representation as the input, while Yadav and Chakraborty (2020) showed that an LSTM trained with domain-specific embeddings as input representations performed better than LASER. Interestingly, some research reported that Machine Learning algorithms such Logistic Regression and Random Forest were able to outperform BERT and even mBERT (Chakravarthi et al., 2020; Javdan et al., 2020). Compared to monolingual text classification, the number of code-mixed datasets is limited. The most notable one is the Spanish-English and Hindi-English code-mixed datasets released for the SemEval 2020 task (Javdan et al., 2020). Other than that, there are code-mixed datasets available between Malayalam-English (Kazhuparambil and Kaushik, 2020), and Tamil-English (Chakravarthi et al., 2020). Interestingly, other than the Spanish-English dataset, all the other datasets we identified involve Indic languages.

1https://github.com/shanakaChathu/ABSA

Bao et al., 2020)
Figure 1: Class distribution of Malayalam-English Dataset

Table 1: Sample of Code-mixed Malayalam-English Dataset

<table>
<thead>
<tr>
<th>Text</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thankyou for lakshmi nair vlogs</td>
<td>Suggestions and queries</td>
</tr>
<tr>
<td>That chopping was ohh veenechi thanku soo much</td>
<td>About the recipe</td>
</tr>
<tr>
<td>Super Tea cake Veena!!! Wl surety try</td>
<td>About the recipe</td>
</tr>
</tbody>
</table>

Table 2: Dataset Details

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>Average Comment Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Si-En</td>
<td>10006</td>
<td>Positive: 61 Negative: 62 Neutral: 85</td>
</tr>
<tr>
<td>Ma-En</td>
<td>4291</td>
<td>Gratitude: 55 About The Recipe: 46 About the Video: 44 Praising: 60 Hybrid: 91 Undefined: 59 Suggestions and Queries: 69</td>
</tr>
</tbody>
</table>

Aspects are specific to the telecommunication domain, as discussed below.

A mini-survey was conducted with the help of 150 users of the telecommunication companies through the social media. First, a detailed list of aspects was identified by analysing user comments. An online form was distributed among social media users and they were asked to select the most important aspects from the aspect list. After that, the highly rated set of aspects was selected as the aspects to be annotated in the corpus. Six aspects, namely network, billing or price, package, customer service, data, and service or product were identified as the final aspects using that mini-survey.

Data was extracted from public forums in Facebook. All these FaceBook pages can be accessed without logging into FaceBook, and are indexed by search engines. All the company names and people names included in the comments were manually removed from the dataset.

Two annotators were employed for the aspect and sentiment annotation. 10000 comments were annotated with the aspects and the sentiment. The sentiment distribution and aspect distribution are shown in the Figure 2 and Figure 3, respectively. Also a sample of the telco data is shown in Table 3. This dataset only contains English and Sinhala written in English script. Overall, about 10% of the corpus is English words.

3 Datasets

3.1 Malayalam-English Dataset

The Malayalam-English dataset was obtained from a food recipe dataset with about 3000 samples Kazzuparambil and Kaushik (2020). Figure 1 and the Table 1 show the class distribution and few samples of the Malayalam-English dataset, respectively. Each comment is tagged with one of the seven classes. Most of the comments in the dataset belong to the undefined class. Comments written in the Malayalam characters were removed from the dataset, and the resulting dataset had 3434 records. Thus the dataset has English words, as well as Malayalam words written in English script. Overall, about 25% of the corpus is English words.

3.2 Sinhala-English Dataset

This dataset was newly created by us. Telecommunication domain has been identified as a low-resource domain. We are not aware of any dataset or research that considered text data in this domain. This dataset was annotated for the following two text classification tasks.

- Document-level sentiment classification, where each user comment is annotated with its sentiment - positive, negative, and neutral.
- Aspect extraction, where each comment is annotated with the aspect term it refers to.
4 Methodology

4.1 Pre-Processing

The Sinhala-English dataset was pre-processed to reduce the noise. Initially, punctuation characters were removed from the text. After that, URLs, mobile phone numbers, and Emails were removed from the text. Also, social media comments contain many emojis in the comments. Those were converted to the word format. The Hashmark was removed from the hashtag as another pre-processing step. After doing all the above-mentioned steps, words were converted to their lowercase form, and the sentences were tokenized. After that basic text pre-processing steps such as converting to lowercase and stop word removal were done to the both datasets.

4.2 Classifying Code-Mixed Social Media Data

In the sentiment analysis problem, only one class is predicted from the positive, negative or neutral classes. But in the aspect prediction mode, more than one class may be predicted if that comment contains more than one aspect. Because of that, sentiment analysis problem was resolved as a multi-class classification problem while the aspect prediction problem was resolved as a multi-label classification problem.

Firstly, in order to setup the baselines, we used recurrent deep learning models on the Sinhala-English dataset. These include RNN, LSTM, GRU, and BiLSTM. fastText and Word2Vec models were trained from a raw corpus of 100000 words extracted from the same sources that were used to create the Sinhala-English annotated dataset. These embeddings were used as the input representation of all these neural models.

Secondly, various improvements were carried out on these models as described below:

- Regularization strategies such as dropout, L1/L2 regularization, and early stopping.
- Integration with CNN models, because CNNs are known to be able to extract more coarse-grained features.
- Stacked models with the aim of extracting rich contextual knowledge from the network’s upper layers. These stacked models contain additional higher layers that extract valuable contextual information from both past and future time sequences (Zhao et al., 2018).

Thirdly, a capsule network was implemented. Capsule networks were selected because they have performed better than LSTM,RNN,Bi-LSTM, etc with Sinhala text classification (not on code-mixed data) (Senevirathne et al., 2020). The capsule network was introduced as an upgrade to CNNs, to be used in NLP applications such as text classification (Sabour et al., 2017). The ability to record context level information in its precise sequence using a vector representation of the capsules is a crucial aspect of the capsule architecture. The dynamic routing mechanism of the capsule network is known to overcome drawbacks of CNNs such as high computational cost and information loss caused by the widely utilized max pooling approach. This basic capsule network architecture...
was combined with LSTM, GRU, BiLSTM, and biGRU models. However, only the combination with the biGRU model gave better results than the recurrent and CNN models, thus only that result will be reported.

Figure 4 shows the Capsule+biGRU network we employed. According to that, firstly raw comments were pre-processed using the text-processing techniques (see Section 4.1). After that, the above trained CBOW word embeddings were used as the first layer of the neural network followed by the Bi-Directional GRU layer. Output from the GRU layer was used in the capsule layer. Finally, a flatten layer was implemented followed by a fully connected Softmax of Sigmoid layer.

Finally, we experimented with the pre-trained contextual embedding models. We experimented with English-BERT and XLM-R. mBERT was not used because it does not include Sinhala. A text classification layer was added on top of both English-BERT and XLM-R models, and they were fine-tuned with the English-Sinhala training data.

To further establish the performance of Capsule+biGRU with respect to English-BERT and XLM-R, these models were tested on the Malayalam-English dataset. Recurrent and CNN models were not tested, as their performance lagged behind the Capsule+biGRU model for Sinhala-English. BERT-uncased model\(^2\) was used as the pre-trained model in all the experiments.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimizer</td>
<td>SGD, RMSprop, Adamax, Adagrad</td>
</tr>
<tr>
<td>Dropout Rate</td>
<td>0.25, 0.50, 0.70</td>
</tr>
<tr>
<td>GRU Activation</td>
<td>relu, tanh, linear</td>
</tr>
<tr>
<td>Number of Capsules</td>
<td>5, 10, 20, 40</td>
</tr>
<tr>
<td>Dimension of Capsules</td>
<td>8, 16, 32, 64</td>
</tr>
<tr>
<td>GRU Length</td>
<td>16, 64, 128, 256</td>
</tr>
</tbody>
</table>

Table 4: Hyper-Parameters of the Capsule+biGRU Network

<table>
<thead>
<tr>
<th>Embedding size</th>
<th>Word2Vec (CBOW)</th>
<th>FastText</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.835</td>
<td>0.824</td>
</tr>
<tr>
<td>200</td>
<td>0.822</td>
<td>0.803</td>
</tr>
<tr>
<td>300</td>
<td>0.839</td>
<td>0.818</td>
</tr>
<tr>
<td>400</td>
<td><strong>0.845</strong></td>
<td>0.817</td>
</tr>
<tr>
<td>500</td>
<td>0.838</td>
<td>0.804</td>
</tr>
</tbody>
</table>

Table 5: Word Embedding Results.

5 Experiments and Evaluation

5.1 Experiment Setup

All experiments were carried out using Google Colab and kaggle. Python was used as the main programming language while Keras was used to build deep neural networks. Three-fold cross-validation was used in every experiment. After that, the best model was trained with the hold-out-based method and hyper-parameter tuning was carried out. Hyper-parameters used in the capsule+biGRU model are shown in the Table 4. The dataset was split into train and validation sets with ratios of 5:1 when doing the hold-out based experiments. Accuracy, precision, recall and F1 were reported as the weighted average for each experiment on the cross-validated dataset.

5.2 Word Embedding Models

The first experiment was to identify the best word embedding models for Sinhala. Both Word2Vec (CBOW) and fastText models were tested for 100, 200, 300, 400 and 500 dimensions. 100000 comments extracted from the same dataset were used to build the word embeddings. CNN model was used as the model to find the best word embedding technique and embedding size, as doing this experiment for all the models is not possible.

According to Table 5, Word2Vec (CBOW) word embedding model with 400 embedding size showed the highest weighted F1 score compared to the

\(^2\)https://huggingface.co/bert-base-uncased
The aspect prediction model also suggested the same thing. Because of that, CBOW with 400 embedding size was used in all experiments.

Table 6 and Table 7 show the results for the two Sinhala-English tasks and the the Malayalam-English task, respectively. Note that the Malayalam-English dataset was used only to compare the Capsule+BiGRU against English-BERT and XLM-R models.

In the sentiment classification task, the Capsule+BiGRU model significantly outperforms English-BERT and XLM-R based solutions. However, the gain in the aspect identification task is not that significant. The result with the Malayalam-English dataset is quite the opposite—the capsule+BiGRU model significantly lags behind English-BERT and XLM-R models.

We can think of multiple reasons for this observation. First and foremost, Sinhala-English dataset was much larger than the Malayalam-English dataset. We believe the number of training samples in the latter dataset was not sufficient to train the Capsule+BiGRU model. In contrast, the pre-trained models could cope with this lack of data. On the other hand, the Malayalam-English dataset had a much higher number of English words compared to the Sinhala-English dataset, which could have been an advantage for the English-BERT model, as well as the XLM-R model that has a significant presence of English. Another reason could be the complexity of the Malayalam-English dataset. Despite the task, XLM-R is consistently lagging marginally behind English-BERT. We attribute this observation to the fact that Sinhala and Malayalam being underrepresented in the XLM-R model. Though we do not know exact size of the commoncrawl corpus used to train the XLM-R model, according to the latest commoncrawl statistics, Sinhala and Malayalam representation was just 0.0070% and 0.0211 %, respectively.

### Conclusion

The objective of this research was to critically analyse the performance of English-BERT and XLM-R models for classifying code-mixed data. We identified that the performance of these models depends on factors such as the size and composition of the code-mixed data. We were able to introduce a novel Capsule+BiGRU model that could outperform the other model.
English-BERT and XLM-R models with a moderate dataset of Sinhala-English 10000 comments (Note in 3-fold cross-validation, about 6600 samples are used for training). This result suggests that, at least for text classification on code-mixed data that involves extremely low-resource languages that are under-represented in the large multilingual embedding models, traditional Deep Learning solutions are still a viability. This research can be considered as one of the very few works that comparatively analysed the performance of these different techniques for code-mixed data with respect to multiple factors and languages. Furthermore, this research publicly released a code-mixed dataset that can be used for two text classification tasks for the extremely low resource language Sinhala. We believe that further research should be conducted with respect to more languages in order to properly determine the impact of the aforementioned factors on classification of code-mixed text.

Acknowledgments

This publication was funded by a Senate Research Committee (SRC) Grant of University of Moratuwa, Sri Lanka.

References


Character-based Thai Word Segmentation with Multiple Attentions

Thodsaporn Chay-intr, Hidetaka Kamigaito, Manabu Okumura
Institute of Innovative Research, Tokyo Institute of Technology
{chayintr@lr., kamigaito@lr., oku@pi.titech.ac.jp

Abstract

Character-based word-segmentation models have been extensively applied to agglutinative languages, including Thai, due to their high performance. These models estimate word boundaries from a character sequence. However, a character unit in sequences has no essential meaning, compared with word, subword, and character cluster units. We propose a Thai word-segmentation model that uses various types of information, including words, subwords, and character clusters, from a character sequence. Our model applies multiple attentions to refine segmentation inferences by estimating the significant relationships among characters and various unit types. The experimental results indicate that our model can outperform other state-of-the-art Thai word-segmentation models.

1 Introduction

Thai running text has neither essential word delimiters nor sentence periods. However, spaces are arbitrarily allowed to separate words, phrases, clauses, and sentences. These characteristics make word segmentation in Thai more difficult than in other languages, such as English, German, and Finnish, which have spaces and periods to identify word and sentence boundaries. Thai word segmentation can be categorized as a sequential-labeling task that assigns a word-boundary label to each character in a fine-grained tagging scheme such as BIES (beginning, inside, end, and singleton) (Xue, 2003), as shown in Figure 1.

Neural-network models have been applied and perform well on character-based Thai word segmentation. Jousimo et al. (2017) applied bidirectional recurrent neural networks with gated recurrent units, while Chormai et al. (2019) proposed an AttaCut, a convolutional-neural-network (CNN)-based model that mainly provides faster and more accurate word inferring motivated by DeepCut (Kitintaranadorn et al., 2019).

Despite the fact that neural-network models with linguistic knowledge can perform almost perfectly, adapting models with pre-trained neural networks, such as word vectors and language models, is still useful (Shao et al., 2018). Seeha et al. (2020) proposed a transfer-learning approach for Thai word segmentation by using a pre-trained character language model for character-based word segmentation. Although it exhibited state-of-the-art performance regarding Thai, it merely uses characters and does not use other information such as word and subword (Sennrich et al., 2016; Kudo, 2018). Additional linguistic knowledge, such as Thai character clusters (CCs) (Theeramunkong et al., 2000) and subword units, has also been successfully used for word segmentation and related tasks (Suntayawalee et al., 2014; Lapjaturapit et al., 2018; Nararatwong et al., 2018; Yang et al., 2019; Li et al., 2019). An attention mechanism (Bahdanau et al., 2015) has been successfully applied to various downstream tasks, particularly a sequence-labeling task (Higashiyama et al., 2019; Tian et al., 2020).

We propose a character-based Thai word-segmentation model with multiple attentions that jointly uses corresponding words and CCs. Our model is based on the bidirectional long short-term

Table 1: Thai word segmentation as a sequence labeling task on BIES tagging scheme

<table>
<thead>
<tr>
<th>B</th>
<th>E</th>
<th>S</th>
<th>B</th>
<th>I</th>
<th>I</th>
<th>I</th>
<th>I</th>
<th>I</th>
<th>I</th>
<th>E</th>
</tr>
</thead>
</table>

*There are three meanings.*

Figure 1: Thai word segmentation as sequence-labeling task on BIES (beginning, inside, end, and singleton) tagging scheme
memory with conditional random field (BiLSTM-CRF) architecture, which is the baseline model for this study, because it has been successfully applied in sequence-labeling tasks for Thai and other languages (Jousimo et al., 2017; Nararatwong et al., 2018; Higashiyama et al., 2019; Seeha et al., 2020; Tian et al., 2020). Our contributions are as follows:

- We use word, subword units, and CCs with multiple attentions to estimate the relationships of characters in character-based Thai word segmentation.
- Our model outperforms the state-of-the-art models in Thai word segmentation, showing the validity of using CCs over subword units.
- Our code will be made publicly available.¹

2 Background and Related Work

2.1 Thai Word Segmentation Revisited

In the early stage of Thai word segmentation, dictionary-based learning techniques had been used along with machine-learning techniques, for instance, Markov models (Kawtrakul and Thumkanon, 1997), decision trees (Sornlertlamvanich et al., 2000; Theeramunkong and Usanavasin, 2001), and CRFs (Haruechaiyasak et al., 2008). CRFs have been shown to be particularly suitable for Thai sequence-labeling tasks (Kruengkrai et al., 2006; Haruechaiyasak and Konyoung, 2009; Kruengkrai et al., 2009; Nararatwong et al., 2018).

In parallel with CRFs, neural-network models, e.g., CNNs (Kittinaradorn et al., 2019; Chormai et al., 2019), LSTM (Treeratpituk, 2017), and BiLSTM (Jousimo et al., 2017), have been applied and performed excellently for character-based Thai word segmentation. Using additional knowledge, such as CC (Lapjaturapit et al., 2018; Nararatwong et al., 2018), transfer learning (Seeha et al., 2020), and stacking ensemble (Limkonchotiwat et al., 2020), along with neural-network models could improve performance.

2.2 Character Clusters in Thai Word Segmentation

Compared with English, the Thai language has various types of characters, i.e., consonants, vowels, tones, and special characters. A word can be formed from a combination of these characters. Thai also has unique linguistic phenomena, for example, some sequential characters tend to be indivisible units. Thus, Theeramunkong et al. (2000) introduced the concept of a CC, which is a set of predefined rules for an indivisible unit on the basis of the Thai writing system.

A CC is smaller than a word but larger than a character. This concept is roughly comparable to a subword unit that is also in the middle of a character and word in terms of length. Using subword units, as well as CCs (Theeramunkong and Tanhermhong, 2004; Sutantayawalee et al., 2014; Lapjaturapit et al., 2018; Nararatwong et al., 2018), in word-segmentation tasks could yield good segmentation performance (Yang et al., 2019; Li et al., 2019). However, decomposing subword units from words requires appropriate parameters and training data, while CCs provide a greater advantage by not requiring any additional parameters. A CC helps avoid segmenting that might violate a writing system (Limcharoen et al., 2009), while subword units will likely not thoroughly exploit morphology (Provilkov et al., 2020), which could generate noise and weaken segmentation performance. This generally makes CCs smaller than subword units, as shown in Figure 2, which enables a comparison of zero-shot segmentation from coarse to fine (top-down) information.

2.3 Attention Mechanism

An attention mechanism was initially proposed by Bahdanau et al. (2015) for neural machine translation focusing on proper parts in sentences, particularly long sentences. It has been successfully applied to downstream tasks, including machine translation (Luong et al., 2015; Vaswani et al., 2017), constituency parsing (Kitaev and Klein, 2018), and sequence labeling (Higashiyama et al., 2019).

Table 1: Thai word segmentation as a sequence labelling task on BIES tagging scheme.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>นิ้วมือ</td>
<td>ความหมาย</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>นิ้ว</td>
<td>ความหมาย</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub</td>
<td>นิ้ว</td>
<td>ความหมาย</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC</td>
<td>นิ้ว</td>
<td>ความหมาย</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

“มี ๓ ความหมาย”

“There are three meanings.”

Figure 2: Comparison of zero-shot segmentation results. S, W, Sub, CC, and C indicate segmentation levels of sentence, word, subword, character cluster, and character, respectively.

¹https://github.com/tchayintr/thwcc-attn
3 Proposed Model

Incorporating candidate words with an attention mechanism into the character-based BiLSTM-CRF architecture could yield superb segmentation performance (Higashiyama et al., 2019). An attention mechanism has the advantage of being flexible for use with additional linguistic knowledge such as CCs and subword units. Thus, we use the concept of CC with an attention mechanism in character-based word segmentation, as shown in Figure 3, by extending the BiLSTM-CRF architecture with word attention from Higashiyama et al. (2019).

Our model estimates CC-integrated character vectors $z$ are incorporated on top of word-integrated character vectors $g$, which are almost identical in architecture. We discuss the major components of our model, i.e., the character-embedding layer, word- and CC-embedding layers, BiLSTM layers for character representation, attention integrations with the BiLSTM layers for integrated representations, and CRF layer.

3.1 Character-embedding Layer

Given a sentence $s$ with $n$ characters that can be represented as $x_{1:n} \equiv (x_1, x_2, \ldots, x_n)$, each character $x_i \in x_{1:n}$ is transformed into a character embedding $e_i^c$ of a $d_c$-dimensional vector (Bengio et al., 2003; Collobert et al., 2011) using lookup-table operation. The lookup table is defined as $E^c \in \mathbb{R}^{d_c \times |V_c|}$, where $d_c$ denotes the dimension of embeddings and $V_c$ denotes a character vocabulary.

3.2 Word- and CC-embedding Layers

Using the word embedding layer as an example, let $V_w$ be a word vocabulary.

Given the character sequence $x_{1:n}$, words are searched on the basis of $V_w$ within a maximum word length $K$ of the character subsequence. A candidate word list $\mathcal{W}_x \equiv (w_1, \ldots, w_m)$ of size $K$ with $m$ candidate words is then obtained, as shown in Figure 3. Each word $w_j \in \mathcal{W}_x \subseteq V_w$ is transformed into a word embedding $e^w$ of a $d_w$-dimensional vector. The word-embedding matrix is defined as $E^w \in \mathbb{R}^{d_w \times |V_w|}$, where $d_w$ denotes the dimension of embeddings. This procedure is also applied to obtain a candidate CC list $\mathcal{C}_x$, which is transformed into a CC-embedding layer $e^c$ of a $d_{cc}$-dimensional vector. The CC-embedding matrix is defined as $E^c \in \mathbb{R}^{d_{cc} \times |V_{cc}|}$, where $d_{cc}$ denotes the dimension of embeddings and $V_{cc}$ denotes a CC vocabulary.

3.3 BiLSTM Layers for Character Representation

The character embedding sequence $e_{1:n}$ is provided to the BiLSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) layers to contextually acquire character context vectors $h_{1:n}$.

A current character context vector $h_i^l \in h_{1:n}$ of the $l$-th layer BiLSTM can be computed bidirectionally:

$$h_i^l = \text{BiLSTM}(h_{i-1}^l, i)$$

$$\equiv \text{LSTM}_f(h_{i-1}^l, i) \oplus \text{LSTM}_b(h_{n-1}^l, n - i - 1),$$

where $h_0^{1:n} = e_{1:n}$, LSTM$_f$ denotes forward LSTM, LSTM$_b$ denotes backward LSTM, $\oplus$ denotes concatenation, and $h \in \mathbb{R}^{2d_r}$ and $d_r$ are hyperparameters.

3.4 Attention Integrations with BiLSTM Layers for Integrated Representations

We use two attention integrations, including word attention and CC attention, to respectively estimate a word-integrated summary vector $a_i^w$ and
CC-integrated summary vector $\mathbf{a}_{i}^{cc}$ for each character in the character sequence. These integrations, which are equal in architecture, accordingly summarize the relationship among characters, words, and CCs.

We apply the composition function weight concatenation (WCON) (Higashiyama et al., 2019) to estimate both summary vectors. This function produces a word-integrated summary vector on the basis of the relationship between a character and their corresponding candidate words. It also can be used to implicitly produce a CC-integrated summary vector on the basis of the relationship between the character with its corresponding candidates words and candidate CCs.

Starting with word-attention integration, we estimate the word-importance score $u_{ij}^{w}$ and word-attention weight $\alpha_{ij}^{w}$ on the basis of the character context vector $\mathbf{h}_{i}$ and candidate word embedding $\mathbf{e}_{j}$ as

$$u_{ij}^{w} = \mathbf{h}_{i}^{T}W_{a}^{w}\mathbf{e}_{j},$$

$$\alpha_{ij}^{w} = \delta_{ij}\exp(u_{ij}^{w}),$$

where $W_{a}^{w} \in \mathbb{R}^{2d_{c} \times d_{w}}$ denotes a trainable weight matrix and $\delta_{ij} \in \{0, 1\}$ indicates whether character $x_{i}$ is included in candidate word $w_{j}$. The word-integrated summary vector $\mathbf{a}_{i}^{w}$ for character $x_{i}$ can be calculated as

$$\mathbf{a}_{i}^{w} = \text{WCON}^{w}(x_{i}, \{w_{j}\}_{j=1}^{m}) = \bigoplus_{l=1}^{L} \alpha_{i+l}^{w}\mathbf{e}_{i+l}^{w},$$

where $\{w_{j}\} = \mathcal{W}_{w}$. Let $K_w$ be the maximum word length, $L_w = \sum_{k=1}^{K_w} k$, $\bigoplus$ denote concatenation, and $i_{l}$ is the corresponding index of potential word list $\mathcal{W}_{w}$ for character $x_{i}$ that is $\{w_{1}^{l}, \ldots, w_{K_w}^{l}\} \equiv \bigcup_{k=1}^{K_w} \bigcup_{s=-k+1}^{0} \{x_{i+s}\}_{s=0}^{k-1}$. A zero vector is applied to Equation 4 when $w_{l}^{l} \notin \mathcal{V}_{w}$.

We then use the BiLSTM layers for transforming the word-integrated summary vectors $\mathbf{a}_{i}^{w}$ into word-integrated character vectors $\mathbf{g}_{i}$. The operation of word-integrated character vector $\mathbf{g}_{i}$ is computed using the BiLSTM layers on the basis of word-integrated summary vector $\mathbf{a}_{i}^{w}$ with its corresponding character context vector $\mathbf{h}_{i}$ as

$$\mathbf{g}_{i} = \text{BiLSTM}(\mathbf{h}_{i} \oplus \mathbf{a}_{i}^{w}).$$

However, the candidate CCs that correspond to the character are used on top of $\mathbf{g}_{i}$ as

$$u_{ip}^{cc} = \mathbf{g}_{i}^{T}W_{a}^{cc}\mathbf{e}_{i}^{cc},$$

$$\alpha_{ip}^{cc} = \delta_{ip}\exp(u_{ip}^{cc}),$$

where $W_{a}^{cc} \in \mathbb{R}^{d_{c} \times d_{cc}}$ denotes a trainable weight matrix and $\delta_{ip} \in \{0, 1\}$ indicates whether character $x_{i}$ is included in the candidate CC $cc_{p}$. The CC-integrated summary vector $\mathbf{a}_{i}^{cc}$ for character $x_{i}$ can be calculated as

$$\mathbf{a}_{i}^{cc} = \text{WCON}^{cc}(x_{i}, \{cc_{p}\}_{p=1}^{L_{cc}}) = \bigoplus_{l=1}^{L_{cc}} \alpha_{i+l}^{cc}\mathbf{e}_{i+l}^{cc},$$

where $\{cc_{p}\} = \mathcal{CC}_{x}$. Let $K_{cc}$ be the maximum CC length, $L_{cc} = \sum_{k=1}^{K_{cc}} k$, and $i_{l}$ is the corresponding index of potential CC list $\mathcal{CC}_{x}$ for character $x_{i}$, i.e., $\{cc_{i_{1}}, \ldots, cc_{i_{L_{cc}}}\} \equiv \bigcup_{k=1}^{K_{cc}} \bigcup_{s=-k+1}^{0} \{x_{i+s}\}_{s=0}^{k-1}$. A zero vector is applied to Equation 8 when $cc_{l_{i}} \notin \mathcal{V}_{cc}$.

Next, we use additional BiLSTM layers to transform the CC-integrated summary vectors $\mathbf{a}_{i}^{cc}$ into CC-integrated character vectors $\mathbf{z}_{i}$ on the basis of a cluster-integrated summary vector $\mathbf{a}_{i}^{cc}$ and its corresponding word-integrated character vector $\mathbf{g}_{i}$ as

$$\mathbf{z}_{i} = \text{BiLSTM}(\mathbf{g}_{i} \oplus \mathbf{a}_{i}^{cc}).$$

A CRF is finally used to estimate the probability of the optimal label sequences $y$.

### 3.5 CRF Layer

A CRF (Lafferty et al., 2001) along with explicitly considering the correlations between adjacent labels has been successfully applied for sequence-labeling-related tasks (Collobert et al., 2011). Let $A \in \mathbb{R}^{|T| \times |T|}$ be a transition matrix for correlations between adjacent labels, where $T$ denotes a set of all possible label sequences, for instance, $T = \{B, I, E, S\}$. The CC-integrated character vector $\mathbf{z}_{i}$ is transformed into an un-normalized label score $\mathbf{s}_{i}$ of the $|T|$-dimensional vector for character $x_{i}$ as

$$\mathbf{s}_{i} = W_{s}\mathbf{z}_{i} + \mathbf{b}_{s},$$

where $W_{s} \in \mathbb{R}^{|T| \times 4d_{c}}$ denotes a trainable weight matrix, and $\mathbf{b}_{s} \in \mathbb{R}^{|T|}$ denotes a trainable bias. Given the input sequence $x_{1:n}$, the corresponding scores for the label sequence $y_{1:n}$ are computed on the basis of transition matrix $A$ and the segmentation label scores $\mathbf{s}$ as follows:

$$\text{score}(x, y) = \sum_{i=1}^{n} (A_{y_{i-1},y_{i}} + \mathbf{s}_{i}[y_{i}])$$
Table 1: Data lengths of BEST2010 corpus, including length of sentences (S), words (W), vocabulary (V), and characters (Ch). “Ency” denotes encyclopedia.

| Domain | |S| | W| | V| | Ch| |
|---|---|---|---|---|---|---|
| Article | 16732 | 1018907 | 25658 | 4178690 |
| Ency | 50629 | 1041067 | 26466 | 4231447 |
| News | 31225 | 1448604 | 37076 | 6174642 |
| Novel | 50134 | 1522258 | 22062 | 5428205 |
| Total | 148720 | 5030836 | 81688 | 20012984 |

The probability of the label sequence can then be obtained as

$$P(y|x) = \frac{\text{score}(x, y)}{\sum_{y' \in T^n} \text{score}(x, y')}.$$  \hspace{1cm} (12)

We can obtain the optimal label sequence $y^*$ by maximizing the sentence score with the Viterbi algorithm:

$$y^* = \arg \max_{y \in T^n} \text{score}(x, y)$$ \hspace{1cm} (13)

The loss function $\mathcal{L}$ is minimized by back propagation during the training process:

$$\mathcal{L}(x, y) = - \log P(y|x)$$ \hspace{1cm} (14)

4 Experiments

4.1 Dataset

We trained and evaluated several versions of our model on the BEST2010 corpus\(^2\), which is the most well-known Thai word-segmented corpus. It contains 5 million words with 20 million characters. The categories and their data lengths are listed in Table 1. We randomly split this corpus into three sets\(^3\): 80% for a training, 10% for a validation, and 10% for a test.

4.2 Subword-unit Integration

Although subword units have been successfully applied to word-segmentation tasks, they might generate noise that decreases segmentation performance when the word dictionary already exists. Therefore, we conducted a comparison of using either subword units or CCs due to their similarity on the different versions of our proposed model.

Let $V_{sw}$ be a subword vocabulary decomposed from the dataset. We simply replace the CC vocabulary $V_{cc}$ with the decomposed subword vocabulary $V_{sw}$. Thus, a candidate subwords list can be acquired and used for the subword-attention integration by applying Equations 6 and 8.

4.3 Integration Order

Considering the flexibility of attention integration, the integration order in our model can be switched. For instance, our model executes CC-attention integration to estimate the relationship between characters and CCs before word-attention integration. This might affect segmentation performance because each integration provides different knowledge. Thus, we implemented a swapped version of our model (Swap) that switches the integration order for comparing the segmentation performance.

4.4 Compared Models

We evaluated the following models:

• **Baseline** A character-based BiLSTM-CRF architecture.
• **Baseline w/ Word** An extension of Baseline that integrates word attention (BiLSTM-CRF with word attention) (Higashiyama et al., 2019).
• **OURS** Our proposed model that integrates word and CC attentions (BiLSTM-CRF with word and CC attentions), as shown in Figure 3.
• **OURS w/o Word** Our proposed model that removes word attention (BiLSTM-CRF with CC attention).
• **OURS w/o CC w/ Sub** Our proposed model that replaces the CC with various sizes of subword units (800-12,800) (BiLSTM-CRF with word and subword attentions).
• **OURS Swap** Our proposed model that swaps the order of word and CC attentions.
• **OURS w/o CC w/ Sub Swap** OURS w/o CC w/ Sub model that swaps the order of word and subword unit attentions.
• **Others** Reproduced Thai word-segmentation models, including well-known models and the state-of-the-art Thai word-segmentation model (Seeha et al., 2020).

---

\(^2\)https://thailang.nectec.or.th
\(^3\)https://resources.aiat.or.th/thwcc-attn/datasets
We used CCs with publicly available libraries, including Phatthiyaphaibun et al. (2016) and TCC-SEG\(^4\), to build the CC vocabulary \(V_{cc}\). To generate the subword vocabulary \(V_{sw}\), we decomposed raw sentences from the dataset into various sizes of subword units using byte-pair encoding (Sennrich et al., 2016) implemented by SentencePiece (Kudo and Richardson, 2018).

We used the common hyperparameters for training the different versions of our proposed model (hereafter, our models), as shown in Table 2. Dropout (Srivastava et al., 2014) was applied to the BiLSTM layers to avoid overfitting as well as non-recurrent layers (Zaremba et al., 2015). We also optimized the model parameters using the Adam optimizer (Kingma and Ba, 2015). We trained our models up to 20 epochs and chose the best one on the basis of the validation process involving the CoNLL\(^5\) evaluation.

We evaluated our models on the test data by using three evaluation metrics, i.e., CoNLL (word-level evaluation), BIES tagging scheme (character-level evaluation), and Bound (boundary-level evaluation) (Seeha et al., 2020). Although the CoNLL and BIES tagging schemes are often used for evaluating the performance of sequence-labeling tasks, the boundary-level evaluation has been used in Thai word-segmentation evaluations. Thus, we also used the boundary-level evaluation in our experiments. Note that our \(F_1\) scores are based on the micro-averaged \(F_1\) score for all evaluation matrices. We conducted a statistical significance test using paired bootstrap resampling (Koehn, 2004) on our results. We set the resampling size to 100,000 iterations and sample size for each resampling to 10% of the test data.

### 4.5 Main Results\(^6\)

Table 3 illustrates the evaluation results among the compared models.

The best of our models, i.e., OURS, achieved the state-of-the-art performance compared with all the other models. From the statistical significance test results, we concluded that OURS surpasses the state-of-the-art model. However, using subword units could slightly improve its performance on average compared with using CCs. This indicates that CCs are more beneficial than subword units, which might generate further noise. OURS, which uses word and CC information, outperformed Baseline w/ Word, which indicates that CCs can be used to complement linguistic knowledge, particularly word information.

#### 4.6 Analysis

**Subword-integration Performance:** We implemented OURS w/o CC w/ Sub on various vocabulary sizes of subword units, as shown in Table 4. The results indicate that the vocabulary size clearly affects subword-integration performance improvement. Specifically, by providing more subword vocabulary to the model, we could consistently increase the overall performance. It may reach the performance of OURS when the subword vocabulary size is enormous. However, it might be difficult to appropriately determine the vocabulary size that will benefit the model. For instance, OURS w/o CC w/Sub with 12,800 subword tokens (OURS w/o CC w/ Sub12800) failed to improve in performance compared with those with 3,200 and 6,400 subword tokens. Thus, the size of subword vocabulary is a crucial parameter that affects the performance for this model.

We chose the best subword-integration model on the basis of validation performance to compare it with OURS, as shown in Table 3. Both subword and CC integrations tended to act as an additional filter layer for word-integrated character

---

\(^4\)https://github.com/tchayintr/tccseg
\(^5\)https://github.com/spyysalo/conlleval.py
\(^6\)We implemented an additional model that replaces the BiLSTM layers with Transformer layers (Vaswani et al., 2017) in Baseline. However, the results were noticeably lower than all other models. Note that we used the hyperparameters for the Transformer layers by referring to Vaswani et al. (2017).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character-embedding size</td>
<td>128</td>
</tr>
<tr>
<td>BiLSTM layers</td>
<td>2</td>
</tr>
<tr>
<td>BiLSTM hidden size</td>
<td>600</td>
</tr>
<tr>
<td>Mini-batch size</td>
<td>128</td>
</tr>
<tr>
<td>Initial learning rate</td>
<td>0.001</td>
</tr>
<tr>
<td>Recurrent layer dropout rate</td>
<td>0.4</td>
</tr>
<tr>
<td>Word-embedding size</td>
<td>300</td>
</tr>
<tr>
<td>Word-vector dropout rate</td>
<td>0.4</td>
</tr>
<tr>
<td>Maximum word (chunk) length</td>
<td>4</td>
</tr>
<tr>
<td>CC/subword-embedding size</td>
<td>300</td>
</tr>
<tr>
<td>CC/subword-vector dropout rate</td>
<td>0.4</td>
</tr>
<tr>
<td>Maximum CC/subword length</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 2: Common hyperparameters for Baselines and our models (top/middle) with exclusive values for our models (bottom). CC hyperparameters can be applied to subword integration.
Table 3: Comparison among our models, baselines, and Others. Best score for each metric is indicated in bold. Our models were significantly better than state-of-the-art model Thai word segmentation model (underline scores) at p-level < 0.01 in pairwise comparison. All models were evaluated on basis of same dataset division. Scores were obtained from mean of two runs. Ours w/o CC w/ Sub scores were reported on the basis of best validation performance among various subword vocabulary sizes. ○ indicates reproduced Thai word-segmentation models.

Table 4: Results of our subword-integration model (OURS w/o CC w/ Sub) with various subword-vocabulary sizes from 800 to 12,800 tokens. underline indicates model that obtained best score in validation process.

representations and improve segmentation performance. However, OURS outperformed OURS w/o CC w/ Sub for every evaluation matrix. We think the main reason is that subword units contain noise while CCs do not. For example, the unit “ก”, which is included in the subword vocabulary, does not exist in Thai word vocabulary and violates the Thai writing system, whereas CCs will not include this type of unit.

Order-of-integration Performance: We compared the performance of our model when the order of attention integrations are swapped.

Table 3 shows that the swapped models decrease segmentation performance compared with their original models, especially the swapped subword-integration model. We argue that subword integration initially adds noise to the character representations, e.g., a subword unit that does not exist in Thai word vocabulary. Therefore, it is difficult for the model to complement such representations in the word-attention integration afterwards. The swapped CC-integration model slightly decreased in performance compared with subword integration because CC vocabulary consists of smaller units that reflect the Thai writing system and includes no noise information. This indicates that OURS outperformed both swapped models and the word information is the priority knowledge to complement a character representation, whereas fine-grained information, i.e., CCs and subword units, are suitable for use after word information as an additional filter layer.

Case Study: Figure 4 shows examples of segmentation results among four models, i.e., Baseline, Baseline /w Word, OURS, and OURS w/o CC w/ Sub. OURS could perfectly segment the example sentence; however, the other models yielded incorrect results. Specifically, the word “ปิย์” violates the Thai writing system by combining the two consonants “ปี” in the word. We think that CC integration filters this type of violation out of the word-integrated character representations, enabling OURS to outperform the other models.

5 Conclusion

We proposed a character-based Thai word-segmentation model that uses multiple attentions on various types of linguistic knowledge, i.e., words, subwords, and CCs. The best version of our model achieved Thai state-of-the-art performance by using the word attention along with CC attention in the BiLSTM-CRF architecture. Further analysis also indicates that using CC can be more beneficial than using subword units in word-segmentation tasks.
References


Xiaoya Li, Yuxian Meng, Xiaofei Sun, Qinghong Han, Arianna Yuan, and Jiwei Li. 2019. Is word segmentation necessary for deep learning of Chinese representations? In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3242–3252, Florence, Italy. Association for Computational Linguistics.

Piya Limcharoen, Cholwich Nattee, and Thanaruk Theeramunkong. 2009. Thai word segmentation based on g1r parsing technique and word n-gram model. In In Proceedings of the 8th International Symposium on Natural Language Processing.


Are Language-Agnostic Sentence Representations actually Language-Agnostic?

Yu Chen Tania Augustinova
Department of Language Science and Technology
Saarland University, Germany
{yuchen,tania}@coli.uni-saarland.de

Abstract
With the emergence of pre-trained multilingual models, multilingual embeddings have been widely applied in various natural language processing tasks. Language-agnostic models provide a versatile way to convert linguistic units from different languages into a shared vector representation space. The relevant work on multilingual sentence embeddings has reportedly reached low error rate in cross-lingual similarity search tasks. In this paper, we apply the pre-trained embedding models and the cross-lingual similarity search task in diverse scenarios, and observed large discrepancy in results in comparison to the original paper. Our findings on cross-lingual similarity search with different newly constructed multilingual datasets show not only correlation with observable language similarities but also strong influence from factors such as translation paths, which limits the interpretation of the language-agnostic property of the LASER model.

1 Introduction
Multilingual joint embeddings map language units from different languages into the same embedding space in order to make them comparable, which facilitates cross-lingual transfer. Being essential for building NLP models of low resourced languages, such an integrated representation is also useful for cross-lingual tasks like machine translation, especially when multiple languages are involved or there is a lack of appropriate data.

While word embeddings are widely used in NLP tasks, sentence representations become quite important for capturing underlying semantic relations in texts across different languages. Hence, instead of simply pooling word representation together, various neural network methods have proposed to produce more coherent sentence representations. Recent advances in multilingual sentence embedding modeling (Schwenk and Douze, 2017; Feng et al., 2020; Hirota et al., 2020) begin to show strong performance on many multilingual NLP tasks, but it does not always work equally well for all languages. We repeat the cross-lingual similarity search task with LASER (Schwenk and Douze, 2017) with more challenging corpora in order to identify what affects the actual performance. Based on our findings, we propose directions for future development of such models.

2 LASER
LASER (Language-Agnostic SEntence Representations) (Schwenk and Douze, 2017; Artetxe and Schwenk, 2019) is contextualized language model based on a BiLSTM encoder trained using a translation objective on parallel data from Europarl (Koehn, 2005), United Nations (Ziemski et al., 2016), OpenSubtitles2018 (Lison et al., 2018), Global Voices (Prokopidis et al., 2016), Tanzil and Tatoeba mostly available on the OPUS website (Tiedemann, 2012). The LASER model is able to handle 93 different languages.

Also Schwenk and Douze (2017) has proposed a similarity-search-based framework in order to evaluate multilingual joint representations. With a collection of $S$ parallel sentences for a given language pair, a multilingual similarity search is performed for the closest target sentence for each of the source sentences, and an error is counted if it is not the reference translation of that sentence in the target language. This approach requires calculating $S^2$ distance metrics. Duplicate sentences need to be removed from the experiment, otherwise the error rates are senseless. In order to have a meaningful comparison across $N$ languages, a similarity search must in addition be performed on an $N$-way parallel sentence set. As the similarity search mainly evaluates the multilingual closeness prop-
erty of multilingual joint sentence embeddings, the representations of the same sentence for different languages should be as similar as possible within the joint representation space.

<table>
<thead>
<tr>
<th>Src</th>
<th>Target language</th>
<th>cs</th>
<th>de</th>
<th>en</th>
<th>es</th>
<th>fr</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs</td>
<td>0.70</td>
<td>0.90</td>
<td>0.67</td>
<td>0.77</td>
<td></td>
<td></td>
<td>0.76</td>
</tr>
<tr>
<td>de</td>
<td>0.83</td>
<td>1.17</td>
<td>0.90</td>
<td>1.03</td>
<td></td>
<td></td>
<td>0.98</td>
</tr>
<tr>
<td>en</td>
<td>0.93</td>
<td>1.27</td>
<td>0.83</td>
<td>1.07</td>
<td></td>
<td></td>
<td>1.02</td>
</tr>
<tr>
<td>es</td>
<td>0.53</td>
<td>0.77</td>
<td>0.97</td>
<td>0.57</td>
<td></td>
<td></td>
<td>0.71</td>
</tr>
<tr>
<td>fr</td>
<td>0.50</td>
<td>0.90</td>
<td>1.13</td>
<td>0.60</td>
<td></td>
<td></td>
<td>0.78</td>
</tr>
<tr>
<td>Avg</td>
<td>0.70</td>
<td>0.91</td>
<td>1.04</td>
<td>0.75</td>
<td>0.86</td>
<td></td>
<td>1.06</td>
</tr>
</tbody>
</table>

Table 1: Pairwise error rates (%) of similarity search of 5 languages (WMT2012).

Table 1 gives detailed similarity search error rates of LASER on the news test set from WMT2012\(^1\). The set consists of 3003 N-way parallel sentences in 5 European languages. Despite the significant differences between these languages, the error rates vary only slightly from the average of 1.06 with the highest error rate at 1.27. The results are consistent with those previously reported in (Artetxe and Schwenk, 2019), i.e. with the base for claiming that the model is language agnostic.

Being able to process 93 languages within a unified framework, supposedly without bias, LASER clearly has limitations. The evaluations have been executed with multilingual data that is either in-domain or fairly close to domains of training data. The applications have focused on a small subset of relatively high resource languages. Also the training data consist of more informational controlled translations, like official documents. It is unclear whether the agnostic property still holds for new domains, for different genres or for all language. After all, languages and translations in reality are much more diverse and robust than the available parallel corpora.

Therefore, we conduct a series of evaluations to examine the framework from different angles, to understand its disadvantages and to find paths for future development.

3 Similarity Search on Multilingual Corpora

We apply the LASER toolkit to two multilingual corpora that are not part of the training data used to build the pre-trained embedding models: the TED corpus (Cettolo et al., 2012) and the appropriate part of the Russian National Corpus (RNC) (Apresjan et al., 2006).

3.1 TED

We first perform a cross-lingual similarity search with the TED corpus (Cettolo et al., 2012), which contains about 17 thousand transcripts, corresponding to around 1000 English talks into 80 languages. As the distribution of translations over these 80 languages is not even and the similarity search requires N-way parallel corpus, we only consider a set of 23 languages (253 possible pairs). After excluding duplicates and limiting the sentence length to 50 tokens, we extract 10 thousand sentences that are 23-way parallel.

TED was not included for training LASER sentence encoders, while covering a large subset of languages that are supported by LASER. Unlike Europarl and UN corpora from documents mainly in the parliament and public office domain, TED involves larger varieties of domains and topics. Based on transcriptions of public presentations, TED corpus is style-wise closer to Europarl, but still differs from the parliament debates in many ways.

Table 2 displays the detailed search error scores (in percent) for 272 out of 506 language pairs in total. The results are quite different from those reported in Section 2 (Table 1) and in (Schwenk and Douze, 2017). The overall error rate for 23 languages is 18.54, almost 9 times as much as the previous ones. Korean and Chinese turned out to be clear outliers among all languages, with search error rates as high as 43.59, while language pairs involving English tend to have less errors than the rest.

Overall, the search error rates in the table appear to correlate with language similarities between the languages. For example, among all language pairs from and to Ukrainian, which is a relatively more difficult language for the cross-lingual search task in general, the two pairs with Bulgarian and Russian perform significantly better than the rest of the language pairs. We observed similar results for Italian to French and Spanish.

It is quite clear that such multilingual sentence representation models are not equally applicable for all language pairs.

---

\(^1\)https://github.com/facebookresearch/LASER/
3.2 Russian National Corpus

To further explore LASER models, we apply them to the Russian National Corpus (RNC) (Apresyan et al., 2006), the multilingual section of which includes literary translations of several classical novels into different languages, which makes the RNC fundamentally different from many other parallel corpora. This is because in addition to rendering the information for the reader, a literary translation also needs to recreate the artistic imagery of the respective original work. The translator must produce a rendition in the target language, taking into account various specific features of the text, sometimes even rewriting it completely. The translations in the RNC are mostly in Slavic languages, some of which are considered low resourced for multilingual NLP tasks. Another distinctive feature of the multilingual section of the RNC is that for some of the novels there are multiple translation into the same language.

The texts in the multilingual RNC are all paragraph-aligned. We segmented the paragraphs into sentences and then align them pairwise with Hunalign (Varga et al., 2007). Then, the pairwise alignments are intersected with a relatively high alignment confidence threshold to produce a N-way parallel set. Unaligned sentences and duplicates are removed. We describe a few experiments with the multilingual RNC in the following subsections.

### 3.2.1 “The Little Prince”

Error rates for cross-lingual similarity search performed on the French novel “The Little Prince” in 12 languages are listed in Table 3. There are 867 sentences for each of the languages.

Similar to English in Table 2, French, the language of the original work, corresponds to significantly lower error rates except for between French and Russian. As a matter of fact, Russian tends to have higher error rates in this experiment. It is possibly due to the fact the Russian translator Nora Gal, a primarily English-Russian translator, could well have been influenced by other versions of the novel.

Furthermore, for each source language, the highest error rate can be more than 10 higher than the lowest and the differences do not seem to be random. The lowest search errors usually happen between languages that are more similar to each other. For instance, with Czech sentence embedding inputs, more Slovak sentences than in any other language are correctly retrieved. Likewise, the Russian-Ukrainian pair exhibits a similar property. The same pattern also exists for Bulgarian, Coati, Macedonian and Serbian. That is, using this matrix of search error rates, we are able to divide the investigated set of Slavic languages into 3 groups:

- Eastern Slavics: Czech, Slovak;
- Western Slavics: Russian, Ukrainian;
- South Slavics: Bulgarian, Coati, Macedonian and Serbian.

In other words, the cross-lingual similarity search task tends to be easier in case of closely related languages. As for the multilingual sentence embedding models, the distances between vectors representing the same sentence in different languages are clearly affected by language similarities. When applying available pre-trained models to other cross-lingual tasks, it is necessary to take into considerations that linguistic distances could affect the performance.
Table 3: Similarity search error rates (%) on “The Little Prince”

<table>
<thead>
<tr>
<th>Src.</th>
<th>bg</th>
<th>cs</th>
<th>fr</th>
<th>hr</th>
<th>mk</th>
<th>ru</th>
<th>sk</th>
<th>sl</th>
<th>sr</th>
<th>uk</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>bg</td>
<td>11.53</td>
<td>9.23</td>
<td>10.38</td>
<td>10.15</td>
<td>15.92</td>
<td>11.19</td>
<td>11.88</td>
<td>10.96</td>
<td>12.34</td>
<td>11.51</td>
<td></td>
</tr>
<tr>
<td>cs</td>
<td>10.73</td>
<td>11.3</td>
<td>11.76</td>
<td>12.57</td>
<td>18.8</td>
<td>9.69</td>
<td>12</td>
<td>13.49</td>
<td>14.42</td>
<td>12.75</td>
<td></td>
</tr>
<tr>
<td>fr</td>
<td>8.65</td>
<td>9.8</td>
<td>9.46</td>
<td>8.3</td>
<td>16.49</td>
<td>10.96</td>
<td>11.07</td>
<td>9.34</td>
<td>12.8</td>
<td>10.76</td>
<td></td>
</tr>
<tr>
<td>ru</td>
<td>15.57</td>
<td>17.88</td>
<td>17.42</td>
<td>17.99</td>
<td>19.72</td>
<td>19.38</td>
<td>18.8</td>
<td>19.49</td>
<td>13.49</td>
<td>17.75</td>
<td></td>
</tr>
<tr>
<td>sk</td>
<td>10.61</td>
<td>8.77</td>
<td>10.5</td>
<td>10.73</td>
<td>12.11</td>
<td>18.57</td>
<td>13.03</td>
<td>12.46</td>
<td>14.53</td>
<td>12.37</td>
<td></td>
</tr>
<tr>
<td>sl</td>
<td>12.34</td>
<td>12.57</td>
<td>12.23</td>
<td>14.76</td>
<td>12.8</td>
<td>20.07</td>
<td>15.8</td>
<td>15.46</td>
<td>16.61</td>
<td>14.74</td>
<td></td>
</tr>
<tr>
<td>sr</td>
<td>10.73</td>
<td>13.26</td>
<td>10.15</td>
<td>8.77</td>
<td>11.65</td>
<td>19.72</td>
<td>12.69</td>
<td>14.65</td>
<td>14.65</td>
<td>12.91</td>
<td></td>
</tr>
</tbody>
</table>

3.2.2 “Alice in Wonderland”

For some of the classical novels, the RNC includes more than one edition for the same language. “Alice in Wonderland” is one of them: there are 3 Russian translations and 2 Polish translations. The respective publishing dates and translators are not provided in the corpus. From the multilingual section of the RNC, we extracted 356 sentences that are parallel in 11 languages and repeated the cross-lingual similarity search on these sentences. Table 4 shows partial results from the experiment.

The novel is originally written in English. While the scores concerning English are frequently lower than expected, there are clear exceptions. When using sentence embeddings generated from two alternative Polish translations to retrieve original English sentences, the error rates range from 5.06% to 23.88%, almost a 400% increase. Apparently, one version (pl2) consists of sentences that are closer to the English original than the other version when mapped to the joint multilingual sentence representation space. The same holds for the three Russian translations. In the English target column, we can see that the three scores related to Russian alternative translations are about 10% apart. Interestingly, the version (ru4) has lower error rates if we search for Ukrainian sentences rather than for English sentences. This brings us to speculate that this Russian version might have served as a source for producing the Ukrainian translation instead of English.

Obviously, there are further factors affecting the language independence of the multilingual sentence representations. The ontological differences from original untranslated texts and translation path appear to keep translations distinguishable from the originals. The literacy translators’ freedom to recreate the work in a new language somehow amplifies the issue. Different translations may differentiate from the original source in so many different ways, This is clearly an interest field to explore with multilingual representations.

3.2.3 Belorussian and Ukrainian

After carrying out the similarity search experiments with all 9 novels in RNC, we summarize the error rates by language in Table 5. The columns “A”–“I” represent the nine novels. Most of the languages have an average error rate around 20%, but Ukrainian and Belorussian has much higher error. In particular, the search error rates of Belorussian experiments rise up to 73.60%. In addition to their distinctive linguistic features, it is likely due to the relatively smaller amount of parallel data that is available for training the LASER models, which are not effective for all languages.

4 Similarity search on different translation paths

As we have discussed in previous sections, the majority of multilingual parallel corpora are collections of translations of the same source documents into different target languages, between which cross-lingual similarity search appears to be more difficult. To investigate the underlying factors that affect similarity search by sentence embeddings, we construct new 4-way parallel data by adding translations from different paths to an existing 3 way parallel set in the following manner: we first select 8 TED talks from the recent online release so as to minimize the translator’s prior knowledge of the talks. All the talks have been transcribed in English and translated into both German and Chinese. There are 629 sentences for each of the 3 languages. The texts are sentence aligned across all 3 languages manually. We sent the sentence segmented German translations as source documents to 4 German-Chinese profes-
Table 4: Similarity search error rates (%) on “Alice in Wonderland”

<table>
<thead>
<tr>
<th>Target language</th>
<th>en</th>
<th>pl₁</th>
<th>pl₂</th>
<th>ru₁</th>
<th>ru₂</th>
<th>ru₃</th>
<th>uk</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>en</td>
<td>25.56</td>
<td>6.18</td>
<td>25.84</td>
<td>10.96</td>
<td>31.18</td>
<td>32.02</td>
<td>23.60</td>
<td>20.55</td>
</tr>
<tr>
<td>pl₁</td>
<td>23.88</td>
<td>18.26</td>
<td>33.99</td>
<td>28.37</td>
<td>38.20</td>
<td>32.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pl₂</td>
<td>5.06</td>
<td>20.22</td>
<td>22.19</td>
<td>14.33</td>
<td>32.87</td>
<td>23.31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ru₁</td>
<td>21.07</td>
<td>34.83</td>
<td>20.79</td>
<td>21.63</td>
<td>33.99</td>
<td>23.88</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ru₂</td>
<td>10.39</td>
<td>28.93</td>
<td>12.64</td>
<td>21.35</td>
<td>27.25</td>
<td>21.91</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ru₃</td>
<td>28.93</td>
<td>38.20</td>
<td>28.37</td>
<td>35.67</td>
<td>29.78</td>
<td>24.72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uk</td>
<td>20.22</td>
<td>30.34</td>
<td>18.54</td>
<td>23.31</td>
<td>21.07</td>
<td>24.16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg.</td>
<td>18.26</td>
<td>29.68</td>
<td>17.46</td>
<td>27.06</td>
<td>21.02</td>
<td>31.28</td>
<td>24.91</td>
<td>24.24</td>
</tr>
</tbody>
</table>

Table 5: Average search error rates by language on Russian National Corpus

<table>
<thead>
<tr>
<th>Target language</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>be</td>
<td>63.40</td>
<td>44.75</td>
<td>50.05</td>
<td>73.60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57.95</td>
</tr>
<tr>
<td>bg</td>
<td>24.29</td>
<td>8.65</td>
<td>14.24</td>
<td>20.33</td>
<td>30.05</td>
<td>32.46</td>
<td>6.94</td>
<td>14.49</td>
<td></td>
<td>18.93</td>
</tr>
<tr>
<td>cs</td>
<td>25.41</td>
<td>16.63</td>
<td>15.76</td>
<td>16.65</td>
<td>30.05</td>
<td>14.31</td>
<td>7.64</td>
<td></td>
<td></td>
<td>18.06</td>
</tr>
<tr>
<td>hr</td>
<td>28.34</td>
<td>8.01</td>
<td>14.35</td>
<td>14.37</td>
<td>30.77</td>
<td>13.01</td>
<td></td>
<td></td>
<td></td>
<td>18.14</td>
</tr>
<tr>
<td>mk</td>
<td>22.24</td>
<td>8.43</td>
<td>15.80</td>
<td>14.93</td>
<td>30.04</td>
<td>13.36</td>
<td></td>
<td></td>
<td></td>
<td>17.47</td>
</tr>
<tr>
<td>pl</td>
<td>27.98</td>
<td>12.33</td>
<td>18.37</td>
<td>19.05</td>
<td>32.44</td>
<td>33.06</td>
<td>15.21</td>
<td>9.63</td>
<td>20.03</td>
<td>20.90</td>
</tr>
<tr>
<td>ru</td>
<td>30.97</td>
<td>7.35</td>
<td>19.72</td>
<td>23.33</td>
<td>41.54</td>
<td>33.85</td>
<td>16.50</td>
<td>14.61</td>
<td>20.66</td>
<td>23.17</td>
</tr>
<tr>
<td>sk</td>
<td>23.49</td>
<td>11.94</td>
<td>15.27</td>
<td>17.52</td>
<td>28.91</td>
<td></td>
<td>6.76</td>
<td></td>
<td></td>
<td>17.32</td>
</tr>
<tr>
<td>sl</td>
<td>21.75</td>
<td>9.97</td>
<td>17.90</td>
<td>16.86</td>
<td>33.70</td>
<td>30.18</td>
<td></td>
<td></td>
<td></td>
<td>21.73</td>
</tr>
<tr>
<td>sr</td>
<td>25.65</td>
<td>10.75</td>
<td>15.80</td>
<td>14.28</td>
<td>29.90</td>
<td></td>
<td>6.13</td>
<td></td>
<td></td>
<td>17.09</td>
</tr>
<tr>
<td>uk</td>
<td>28.61</td>
<td>8.74</td>
<td>16.91</td>
<td>20.84</td>
<td>38.81</td>
<td>36.41</td>
<td></td>
<td></td>
<td></td>
<td>25.05</td>
</tr>
</tbody>
</table>

Table 6: Cross-lingual similarity search error rates (%) on a 4-way parallel corpus that contains translations produced from different paths

<table>
<thead>
<tr>
<th>Target Language</th>
<th>(en-)de</th>
<th>(de-)zh en org.</th>
<th>(en-)zh</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(en-)de</td>
<td>2.70</td>
<td>8.74</td>
<td>10.81</td>
<td>7.42</td>
</tr>
<tr>
<td>(de-)zh</td>
<td>2.38</td>
<td>11.45</td>
<td>12.72</td>
<td>8.85</td>
</tr>
<tr>
<td>en</td>
<td>9.86</td>
<td>11.92</td>
<td>8.74</td>
<td>10.17</td>
</tr>
<tr>
<td>(en-)zh</td>
<td>13.04</td>
<td>14.31</td>
<td>7.15</td>
<td>11.50</td>
</tr>
<tr>
<td>Avg.</td>
<td>8.43</td>
<td>9.64</td>
<td>9.11</td>
<td>10.76</td>
</tr>
</tbody>
</table>

Table 283
sional translators and asking them to produce Chinese translations and to stay close to the German texts semantically as much as possible. Accordingly, the translators might not have as much creative space to improvise in the process as a normal freelance translator. Since the German texts have been sentence segmented before the translation, the resulting Chinese texts can be easily aligned back to the 3-way parallel set after review. The newly translated Chinese texts are then added into the set as yet another separate copy, and we perform cross-lingual similarity search on the updated set using LASER as shown in Table 6.

This data set includes two Chinese versions of the same English presentations that may be considered paraphrases to each other: one is directly translated from the original English transcription and the other is pivotally translated through German. In this setup, a perfect language agnostic embedding model should be able to map the sentences into vector clusters, each of the clusters representing an English sentence with its German translations together with two variants of Chinese translations. However, our results contradict this assumption.

The distance between the English original and the English-Chinese translations is not far from that between the English and the German ones. The differences in error rate are within 1, around 6 sentences out of 629. The German-Chinese translations did turn out to be much closer to the German texts in the sentence representations. We believe it is due to the strict requirements given to the translator. Despite being translations into the same language, the two Chinese texts lead to the highest search error rates.

Figure 1 illustrates an example from this data set. The dots represent vectors generated for each of the sentences with the LASER toolkit in a joint sentence embedding space and the edges connecting the dots are labeled with distances. Notably, the German translation uses period instead of “—” at the end of the sentence, which is clearly not possible to recover in the German-Chinese translation. This is reflected in the distances between these representation vectors. The German-Chinese translation is closer to the German sentence rather than to the English or the Chinese sentence. Similar examples are fairly common in this set, which also explains the distinctive performance of cross-lingual similarity search for the translations into the same language we discussed in Section 3.2. It is inevitable that translation introduces distortions into texts. Even though the ultimate goal of building up a multilingual sentence representation model is to allocate sentences with the same meanings regardless of their languages as close to each other as possible, translation distortions are still visible in the state-of-the-art multilingual sentence representations. Potentially, the joint sentence embeddings may be one way to identify translation paths or even to quantify translation distortions.

5 Conclusion

Neural embeddings have been widely applied in all fields of natural language processing. Multilingual embeddings with shared representation space enable few-shot and zero-shot transfer from one language to another with minimum additional training or data requirement. Recent developments on multilingual sentence representations such as LASER have opened up new path towards competitive NLP performance across high- and low-resource languages.

Yet, our evaluation of LASER reveals many constraints when applied in realistic and challenging scenarios. The performance of the framework is largely influenced by the similarity between languages in the multilingual application. Not all languages work equally well currently. Low performance on specific languages is attributed to the small training data size.

These observations caution the interpretation of language-agnostic property of such cross-lingual sentence representations and their application in multi-lingual NLP applications. The newly constructed multilingual corpus in this paper can be used as a new evaluation benchmark for future cross-lingual representation learning research. We
plan to release the data set to the research community.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 232722074 – SFB 1102.

References


Investigating Dominant Word Order on Universal Dependencies
with Graph Rewriting

Hee-Soo Choi
Sorbonne Université and
Université de Lorraine, CNRS, Inria,
LORIA, F-54000 Nancy, France
hee-soo.choi@loria.fr

Bruno Guillaume
Université de Lorraine, CNRS, Inria,
LORIA, F-54000 Nancy, France
Bruno.Guillaume@loria.fr

Karën Fort
Sorbonne Université / STIH and
Université de Lorraine, CNRS, Inria,
LORIA, F-54000 Nancy, France
karen.fort@loria.fr

Guy Perrier
Université de Lorraine, CNRS, Inria,
LORIA, F-54000 Nancy, France
Guy.Perrier@loria.fr

Abstract
This paper details experiments we performed on the Universal Dependencies 2.7 corpora in order to investigate the dominant word order in the available languages. For this purpose, we used a graph rewriting tool, GREW, which allowed us to go beyond the surface annotations and identify the implicit subjects. We first measured the distribution of the six different word orders (SVO, SOV, VSO, VOS, OVS, OSV) in the corpora and investigated when there was a significant difference in the corpora within a given language. Then, we compared the obtained results with information provided in the WALS database (Dryer and Haspelmath, 2013) and in Östling (2015). Finally, we examined the impact of using a graph rewriting tool for this task. The tools and resources used for this research are all freely available.

1 Introduction
Language typology has proven to be useful in natural language processing (NLP) (Bender, 2016; O’Horan et al., 2016), for example for improving performance in language transfer (Naseem et al., 2012; Ahmad et al., 2019) and joint learning.

As noted by O’Horan et al. (2016) “WALS is currently by far the most commonly-used typological resource in NLP due to its broad coverage of features and languages”. However, the WALS database (Dryer and Haspelmath, 2013) has been compiled from the work of 55 linguists1 and is not systemically based on a large quantity of data. Moreover, it does not provide all the considered features for all the languages it covers.

On the other hand, the Universal Dependencies (UD) framework (Nivre et al., 2016) provides a large number of corpora annotated in dependency syntax (in version 2.7, there are 183 corpora for 104 languages).

We decided to automatically extract from the UD corpora one of the most used features in NLP, the dominant word order, i.e. the way the subject (S), verb (V) and object (O) are ordered in a language (feature 81A in WALS). To do so, we use a freely available graph rewriting tool, which allows us to perform complex searches, to take into account the context of the construction and to add or modify the existing annotations to expose relations which are not directly accessible in the corpora.

These experiments led us to define what is a dominant word order, to observe the distribution in word orders within the corpora of a given language, to determine the frequency of the different word orders in all the considered corpora, and to compare the obtained results with those of existing databases, including WALS.

2 Previous Work
2.1 UD-based Typology
Dependency treebanks have already been used to investigate the order of subject and object in different languages. Liu (2010) presented a statistical overview of several binary parameters including SV vs VS, OV vs VO on 20 languages and compared their results with WALS’. However, their experiments were conducted before the UD framework, on treebanks with different annotation schemes.

To our knowledge, the closest work to ours is that of Östling (2015). He considered word order typology based upon the translated and aligned new testament in almost 1,000 languages and compared his results with WALS data. The main difference

---

1See: https://wals.info/author.
with our work is that for us, identifying the dominant word order is our goal and not just a production allowing the evaluation of a system. Besides, the data used in his experiment was generated automatically, rather than (at least partially) manually annotated. The produced data is available, so we were able to compare our results with his.

UD treebanks were also used to study word order freedom. Futrell et al. (2015) and Berdicevskis and Piperski (2020) examined the word order freedom of subjects and objects, focusing on the correlation with case marking.

More recently, Alzetta et al. (2018) applied a plausibility assessment algorithm to the UD treebanks to assess its usability in identifying typological features. They focused on the subject-verb and adjective-noun orders and experimented with three languages, English, Italian and Spanish. While their analysis is quite thorough, the algorithm they employed is not available, so their work cannot be extended.

Finally, Gerdes et al. (2019b) tested some of Greenberg’s universals on UD. Their work does include word order information but focuses on only two classes (the verb is before or after the object/subject). Besides, they decided to merge the treebanks for multi-corpora languages.

### 2.2 Enriching UD Annotations

There is no easy way to decide which dependency relations should be taken into account in order to observe word order dominance. In basic UD annotations, the tree restriction of usual dependency annotation frameworks impose some arbitrary choices: it is not possible to consider that the same token can be used twice as subject of different verbs. In our study, we try to overcome this limitation by making explicit some “syntactic” relations which cannot be expressed in UD (see Section 4.3). In Section 7, we compare what we observe using what we call implicit subjects with the same analysis on basic UD annotations only.

Similar types of enrichment have been proposed before, namely the Enhanced Universal Dependencies and the Deep Universal Dependencies.

Enhanced Universal Dependencies (EUD) were proposed in Schuster and Manning (2016). The goal of this work is to create an annotation which is more suitable for natural language understanding.

The goal of the Deep Universal Dependencies (DUD) (Droganova and Zeman, 2019) is also to provide annotations adapted to natural language understanding. DUD expresses relations that are closer to predicate-argument structure than the annotations of EUD, using relations names (arg1, arg2, ...) borrowed from semantic frameworks like the Abstract Meaning Representation (AMR) (Banarescu et al., 2013). DUD is built automatically from EUD when annotations are available or with an automatic production of EUD for other corpora.³

### 3 Methodology

#### 3.1 Taking the Corpora as Basis

Our study is based on the version 2.7 of UD, with 183 corpora and 104 languages available. Since our experiments consist in the extraction of statistics from data in corpora, we chose to eliminate corpora with fewer than 1,000 sentences, since we consider them too small to be representative of the language. Once this filter was applied, we obtained 141 corpora in 74 languages, which constitute the UD 2.7K corpus.

We decided to compile statistics at the corpus level rather than the language level, in order to observe variations between corpora of a given language and to compare the significance between them. 29 languages are represented by more than one corpus and for the 45 remaining ones we consider that “corpus equals language”.

¹See: https://universaldependencies.org/u/overview/enhanced-syntax.html.
³At the time of writing, DUD annotations are not available for version 2.7.
3.2 Defining a Dominant Word Order

Describing an order as a language’s dominant order can have two meanings: either the order is the only possible one for the language, or the language exhibits several different orders and one is more frequently used.

In our experiments, we count the occurrences of the six possible orders in UD 2.71K, which means that our results are based only on the occurrence frequencies of the orders and therefore depend heavily on the composition of the corpora. Although we are aware of possible biases due to the corpus’s degree of representativeness, our purpose is to determine a dominant order per corpus from raw data and to check whether we obtain results which are consistent with those of descriptive grammars.

Inspired by Dryer (2013), we consider the most frequent order as the dominant order for a given corpus provided that it is at least twice as frequent as the next most frequent. This means that for each corpus, we observe the ratio between the number of occurrences of the most frequent order with respect to the number of occurrences of the second most frequent order; if the ratio is greater than or equal to 2, the most frequent order is the dominant order, else we consider the corpus to be NDO (No Dominant Order). This allows us to classify as NDO corpora exhibiting little differences between two orders (for example, GERMAN-GSD with implicit subjects shows 35.7% SOV and 34.8% SVO). When the results differ among corpora of a given language, we study the corpora on a case by case basis.

3.3 Dealing with UD Specifics

In UD (Nivre et al., 2016), a given label can be ambiguous with respect to syntactic relations. For example, the labels xcomp and ccomp are used for both direct and indirect objects. Because of this limitation, we restrict our study to nominal objects, i.e. to obj relations. A similar difficulty arises with subjects. In UD, a personal subject is annotated with a relation subj, while an impersonal subject is annotated with the relation expl, which is also used for other relations with expletives. This ambiguity leads us to ignore impersonal subjects in our study.

Due to the tree constraint, some relations are not explicitly given in the data. In our study, this can affect subjects that can be shared by several verbs in coordination or through control of raising verbs. We call these hidden relations implicit relations.

For instance, consider the Polish sentence:

Kuba tego nie potrzebuje ale ma to od Kuba this not need but has this from
mamy mom
Kuba does not need this, but has it from her mother

There is an implicit subject relation between ma and Kuba which is not represented in the UD annotation. In our experiments, we ran an extended search on UD data with implicit subjects that can be predicted from surface syntax. Implicit objects also exist but it is not possible to recover them automatically from surface syntax. In the previous example, tego is the object of potrzebuje but it is not possible to determine if tego is an implicit object of ma.

Besides these issues, UD 2.7 includes two code-switching corpora: Turkish-German and Hindi-English. They were added as new “languages” and we therefore consider them here as such.

4 Going Deeper with Graph Rewriting

4.1 GREW

GREW is a graph rewriting tool dedicated to NLP applications, which can be used to query treebanks using graph patterns written with a specified syntax. Given a set of queries and a set of corpora, a script produces a table with the number of occurrences of each query in each corpus (see Section 4.2, for examples). An online interface to the tool is available, which enables users to observe examples in context within corpora and to interactively design and debug the patterns before running the script.

GREW also allows users to describe a set of transformations and to apply them to each item in a corpus. In this paper, we use this feature to enrich the available annotations (see Section 4.3).

4.2 Extraction Patterns

The patterns we use to extract data in UD 2.71K include two syntactic relations: subject and object. As explained in Section 3.3, only nominal
objects can be reliably recovered from UD annotations. To be consistent, we use the same restriction for the subject relation and focus only on nominal subjects (nsubj), without considering clausal subjects (csubj). For instance, the GREW pattern for SVO is presented in Figure 1.

```
pattern {
  V [upos=VERB];
  V -[1=nsubj|isubj]-> S; V -[1=obj]-> O;
  S << V; V << O
}
```

Figure 1: GREW pattern for SVO.

In UD, it is possible to include subtypes in relations, for instance the relation nsubj:pass can be used for a regular nominal subject in a passive construction. However, as these extensions are defined at the language level, we do not consider them here. The GREW syntax 1=subj|isubj allows to capture all relations that are either subj or isubj with or without subtypes.

4.3 Enriching UD Annotations

When used on UD annotations, the aforementioned extraction patterns present some limits as they only identify cases where the subject and the object are syntactically directly related to the same verb. However, there exist constructions admitting a subject and an object with two different governors. In our study, we consider two cases where the information can safely be recovered from surface annotations by adding implicit subjects, isubj, in an enriched UD annotation (see Section 3.3).8

The first case is coordination: when two clauses involving the same subject are linked by a coordinating conjunction with an ellipsis of the subject to the head of the second clause. More technically, this is described by the rule in Figure 2: if two verbs V1 and V2 are linked by a conj relation and V2 does not have its own subject; then add the subject S1 of V1 as an isubj of V2. For instance in a sentence “He obtains these things, but loses the ability to manage them,” a relation isubj will be added from loses to He.

The second case we consider is control or raising. In UD, this is annotated with the relation xcomp between the two verbs. We can use a rule that is similar to the one in Figure 2 with xcomp instead of conj. In the sentence “I should like to address

```rule conj {
  pattern {
    V1 [upos=VERB]; V2 [upos=VERB];
    V1 -[1=conj]-> V2;
    V1 -[1=nsubj]-> S1;
  }
  without { V2 -[1=nsubj]-> S2; }
  commands { add_edge V2 -[isubj]-> S1; }
}
```

Figure 2: GREW rule adding the isubj relation.

one final point.”, the enriched annotation will show a relation isubj from address to I.

5 Determining Dominant Word Order in Multi-Corpora Languages

We detail here the results obtained for multcorpora languages. For the mono-corpus languages, we examine our results as compared to WAL$S$ and Östling (2015) in Section 6.

5.1 Intra-language Consistency

We obtain the number of occurrences of each of the six possible orders for each corpus in UD 2.71K. This data can be used to determine whether different corpora of a given language exhibit similar distributions. For this purpose, we compute the cosine between the 6-dimensional vectors for each corpus. This technique of comparing two feature vectors as a means of comparing two languages has already been used in several works on language typology (Georgi et al., 2010; Berzak et al., 2014). We expect two corpora of the same language to display similar distributions and therefore expect a cosine value close to 1.

The lowest value we observe is for two corpora of Romanian. Table 1 illustrates the vectors describing the distribution of the six possible orders for the three Romanian treebanks and Figure 3 represents as a heatmap the cosine values between these vectors.

Figure 4 reports the minimum cosine value among all possible pairs of corpora for the 29 multcorpora languages.

Ten languages have a value below 0.95 and three have a value below 0.8 (Romanian, Hindi, Arabic). We present below a basic analysis of these results, either by seeking an explanation in the description of the corpora on the UD website or by asking language experts to examine the data.

Different text genres Four languages present corpora in different text genres, which could
Table 1: Distribution vectors for the Romanian treebanks.

<table>
<thead>
<tr>
<th>Romanian_NONSTANDARD</th>
<th>SVO</th>
<th>SOV</th>
<th>VSO</th>
<th>VOS</th>
<th>OSV</th>
<th>OVS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>38.07%</td>
<td>31.87%</td>
<td>9.66%</td>
<td>3.97%</td>
<td>1.71%</td>
<td>14.72%</td>
</tr>
<tr>
<td>Romanian_RRT</td>
<td>85.32%</td>
<td>7.76%</td>
<td>1.12%</td>
<td>0.70%</td>
<td>1.18%</td>
<td>3.91%</td>
</tr>
<tr>
<td>Romanian_SIMoNERo</td>
<td>97.61%</td>
<td>0.97%</td>
<td>0.09%</td>
<td>0.09%</td>
<td>0.13%</td>
<td>1.10%</td>
</tr>
</tbody>
</table>

Figure 3: Cosine values between the three Romanian corpora in UD 2.7.

Figure 4: Multi-corpora (number in parenthesis) languages ordered by minimum cosine value.

explain the low cosine value: Dutch (0.928), French (0.912), Romanian (0.729) and Slovenian (0.947). One corpus in Romanian (Romanian_NONSTANDARD) is dedicated to non-standard usage of that language (see Figure 3). Some corpora focus on specific types of texts: questions (French-FQB, which clearly stands out in Figure 5) and/or material from test suites and sentences from a reference grammar (Dutch-ALPINO). French and Slovenian present corpora of spoken language (French-SPOKEN and Slovenian-SSL). In Czech (0.925), one of the corpora (Czech-FicTree) contains only fiction and shows a higher proportion of SOV, while the four other Czech treebanks are clearly SVO.

Different text periods For two dead languages (Latin and Ancient Greek), corpora gather texts from very different historical periods, which could explain the differences. Latin texts range from 1st century BC (Latin-PERSEUS) to 13th century Medieval Latin (Latin-ITTB). For Ancient Greek, very different kinds of text are mixed: Ancient.Greek-PROIEL contains both Herodotus (5th century BC) and Bible texts; the other corpus (Ancient.Greek-PERSEUS) is a larger mix of several periods from Homer (8th century BC) to Athenaeus (late second century). Undoubtedly, the fact that Latin and Ancient Greek are considered free word order languages amplifies the phenomenon. As for German, two corpora are NDO and two are SOV, German-HDT with a low ratio (2.01) and German-LIT. The latter is the only corpus to be composed of 18th century texts.

---

9 The cosine between these two subcorpora is 0.907.
Non-standard annotations In one of the two Hindi corpora (HINDI-HDTB), there is a large percentage of SVO cases (82.5%) where the object is a verb, in contradiction of the UD guidelines. If we consider only nominal subjects and objects in the patterns, the cosine value rises to 0.993.

Language specifics In Arabic (modern standard), the basic order is VSO. However, SVO is used in cases of topicalization of the subject and in completives. The PADT corpus contains many titles of news articles featuring topicalization, which could explain the prevalence of SVO.

5.2 Dominant Word Order in Multi-Corpora Languages

For all multi-corpora languages with a minimum cosine value above 0.95, the dominant word order ratio consistently produces the same dominant order for all corpora of the language, except for Estonian which presents a SVO corpus (ESTONIAN-EDT) and a NDO corpus (ESTONIAN-EWT), corresponding to different text genres (fiction, news, nonfiction, academic vs blog, web, social). 14 multi-corpora languages are thus identified as SVO (Chinese, English, Faroese, Finnish, Galician, Icelandic, Indonesian, Italian, Norwegian, Polish, Portuguese, Russian, Spanish and Swedish) and four multi-corpora languages as SOV (Japanese, Korean, Persian and Turkish).

Out of the 10 multi-corpora languages with a minimum cosine value below 0.95, two present a clear dominant order SVO: French and Czech. As for Dutch and Ancient Greek, they both are NDO, but with inconsistent main orders: SOV/SVO and SVO/SOV. The six other languages present inconsistent dominant word orders. However, Romanian and Slovenian are both SVO in their standard or written form, even though one of their corpora is NDO (SVO/SOV). As for German, two out of four corpora are NDO (SOV/SVO) and two corpora are SOV. However, this result can be attributed to a threshold effect, since these two corpora present a SOV order at low ratios (2.01 for GERMAN-HDT, 2.53 for GERMAN-LIT).

Two corpora of Arabic are NDO (one is SVO/VSO and the other VSO/SVO) and one corpus is VSO. For the reasons explained in Section 5.1, we consider that the dominant order is most probably VSO. Hindi has one SOV and one NDO (SOV/SVO) corpus, but if we remove SVO occurrences probably due to annotation errors (i.e. O is a verb) in the latter, both corpora are clearly SOV. Latin is the language with the most heterogeneous corpora, with three NDO corpora (one SVO/SOV, two SOV/SVO) and one SOV corpus. These differences can probably be explained by the time range among texts.

Regarding the two code-switching languages, Hindi-English is considered NDO (SVO/SOV) which is consistent since English is SVO and Hindi SOV. As for Turkish-German, the corpus presents a SOV order, Turkish and German having this order in common.
6 Comparison with other Sources

Amongst the 74 languages available in UD 2.71K, WALS does not cover the seven dead languages, nor the two code-switching “languages”. In addition, WALS does not provide Feature 81A for six languages. In Östling (2015), 22 languages are not in the database and seven are in neither sources, Galician, Hindi-English, Turkish-German and four dead languages (Old French, Old Russian, Sanskrit, Akkadian).

6.1 Differences with WALS

<table>
<thead>
<tr>
<th>Language</th>
<th>UD 2.71K</th>
<th>WALS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amharic</td>
<td>1 NDO</td>
<td>SOV</td>
</tr>
<tr>
<td>Arabic</td>
<td>1 VSO, 2 NDO</td>
<td>VSO</td>
</tr>
<tr>
<td>Belarusian</td>
<td>1 SOV</td>
<td>NDO</td>
</tr>
<tr>
<td>Estonian</td>
<td>1 SOV, 1 NDO</td>
<td>SVO</td>
</tr>
<tr>
<td>German</td>
<td>2 SOV, 2 NDO</td>
<td>NDO</td>
</tr>
<tr>
<td>Greek</td>
<td>1 SOV</td>
<td>NDO</td>
</tr>
<tr>
<td>Hindi</td>
<td>1 SOV, 1 NDO</td>
<td>SOV</td>
</tr>
<tr>
<td>Mbya Guarani</td>
<td>1 NDO</td>
<td>SVO</td>
</tr>
<tr>
<td>Romanian</td>
<td>2 SOV, 1 NDO</td>
<td>SVO</td>
</tr>
<tr>
<td>Slovenian</td>
<td>1 SOV, 1 NDO</td>
<td>SVO</td>
</tr>
<tr>
<td>Urdu</td>
<td>1 NDO</td>
<td>SOV</td>
</tr>
</tbody>
</table>

Table 2: Differences with WALS (for UD 2.71K, we detail by corpora).

We compare our results with Feature 81A (Order of Subject, Object and Verb) in WALS. We have 59 languages in common and we consistently observe the same dominant word order for 48 of these. In Table 2, we detail the remaining 11 languages where our observations are not fully consistent with WALS classification. Taking into account the explanations in Section 5.2 about multi-corpora languages, we have five languages with one corpus where we disagree with WALS: Amharic, Belarusian, Greek, Mbya Guarani and Urdu.

Belarusian and Greek can be considered relatively free word order languages, hence the NDO order in WALS. In our results, Belarusian is SVO with a ratio of 10.43, however the BELARUSIAN-HSE corpus is based on texts included in the Belarusian-Russian parallel subcorpus of the Russian National Corpus. Russian being a SVO language, this may explain the high proportion of SVO.

Moreover, it is more common to find the SVO order as the basic order in written Belarusian. Similarly, the basic order in Greek being SVO, this may explain the ratio of 7.31 of SVO order in our results.

As for Mbya Guarani and Urdu, the most frequent order corresponds to the order in WALS. Mbya Guarani is NDO (SVO/SOV) with a ratio of 1.25 and Urdu NDO (SOV/SVO) with a ratio of 1.52. Finally, Amharic has an OVS order as the most frequent order, contrary to WALS’.

There are six languages present in WALS which do not have the Feature 81A: Galician, Faroese, Kazakh, Maltese, Naija and Slovak. Our results could therefore be used to enrich WALS’ data.

6.2 Differences with Östling (2015)

<table>
<thead>
<tr>
<th>Language</th>
<th>UD 2.71K</th>
<th>Östling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amharic</td>
<td>1 NDO</td>
<td>SOV</td>
</tr>
<tr>
<td>Ancient Greek</td>
<td>2 NDO</td>
<td>SVO</td>
</tr>
<tr>
<td>Armenian</td>
<td>1 NDO</td>
<td>SVO</td>
</tr>
<tr>
<td>Basque</td>
<td>1 SOV</td>
<td>SVO</td>
</tr>
<tr>
<td>Dutch</td>
<td>2 NDO</td>
<td>SOV</td>
</tr>
<tr>
<td>Estonian</td>
<td>1 SOV, 1 NDO</td>
<td>SVO</td>
</tr>
<tr>
<td>German</td>
<td>2 SOV, 2 NDO</td>
<td>SVO</td>
</tr>
<tr>
<td>Hindi</td>
<td>1 SOV, 1 NDO</td>
<td>SOV</td>
</tr>
<tr>
<td>Hungarian</td>
<td>1 NDO</td>
<td>SVO</td>
</tr>
<tr>
<td>Latin</td>
<td>1 SOV, 3 NDO</td>
<td>SVO</td>
</tr>
<tr>
<td>Mbya Guarani</td>
<td>1 NDO</td>
<td>SVO</td>
</tr>
<tr>
<td>Romanian</td>
<td>2 SOV, 1 NDO</td>
<td>SVO</td>
</tr>
<tr>
<td>Slovenian</td>
<td>1 SOV, 1 NDO</td>
<td>SVO</td>
</tr>
<tr>
<td>Welsh</td>
<td>1 VSO</td>
<td>SVO</td>
</tr>
</tbody>
</table>

Table 3: Differences with Östling (2015) (for UD 2.71K, we detail by corpora).

The data presented in Östling (2015) is computed from the automatically aligned New Testament. The corpora are homogeneous and the data on which the dominant order is computed can sometimes be very small (for Hungarian, 127 structures vs 876 in our experiment). Moreover, Östling (2015) considers the single most prevalent order as the dominant one, without taking into account the difference with the second one.

We have 52 languages in common and observe the same dominant order for 38 of these. Table 3 reports what we observe for the 14 other languages. Out of these languages, 9 NDO languages have the same first order as Östling (2015). The 5 remaining ones (Amharic, Ancient Greek, Basque, Latin and Welsh) present different first orders.

---

10We are aware that WALS and Östling (2015) classifications only deal with transitive clauses, however the UD annotations do not allow us to extract them precisely.

11See: https://wals.info/feature/81A.
7 Influence of Implicit Subjects

As said earlier, we decided to enrich the basic UD annotations, but the choice we made is quite arbitrary. In order to evaluate how this may have impacted our observations, we conducted the same experiment without taking into account the implicit subjects.

Table 4 lists the five languages for which the two experiments predict a different word order for at least one corpus or different first rank in NDO ordering. Adding isubj changes the dominant word order in four corpora: CZECH-FiC TREE, ESTONIAN-EWT, GERMAN-HDT and TURKISH-GERMAN-SAGT. However, we note that in the four cases, one of the two ratios is close to the threshold. We observe an unexpected change for LATIN-LLCT which remains NDO, but with different top ranks (SOV/SVO with isubj and OSV/SVO without isubj). Latin is the language where we see the most important changes as can be observed by comparing the heatmaps in Figure 6.

Table 5 reports the corpora where the two experiments show a high difference (more than 5%) in term of first rank word order prediction. Only the WELSH-CCG corpus has a significant lower first rank with isubj, other corpora with a large difference present an higher first rank when isubj are taken into account. Again, the LATIN-LLCT exhibits a strange behavior with different first rank word order.

8 Conclusions and Perspectives

The main outcome of these experiments is the determination of the dominant word order for 74 languages, based on large amounts of annotated data. This result can be used for NLP applications. On the linguistic side, our findings could be used to reinforce the results published in WALS and complete them in some cases. However, our results differ from WALS’ for 11 languages, and for these, a more thorough analysis should be conducted by specialists of said languages. We are planning to experiment using graph rewriting to explore other universals, like Greenberg’s (Greenberg, 1963) or other missing features in WALS.

Graph rewriting can be used to enrich the UD annotations but it can also be used to reorganise more deeply the tree dependency graph. In Gerdes et al. (2019b), the observations were done on such a deeper reorganisation of the dependency tree structure, proposed in Surface Syntactic Universal Dependency (Gerdes et al., 2019a) which was already produced using GREW-based graph rewriting.

Our experiments can be replicated and extended: all the tools and resources are freely available and we also provide the patterns and scripts to be used\(^{12}\).

Acknowledgements

We thank the reviewers for their useful remarks. We also wish to thank the colleagues who kindly took the time to answer our questions concerning some of the results we obtained in languages we do not speak: Sashi Narayan and Lydie Lemoine for Hindi, Hilda Mock for Arabic, Kim Gerdes for German and Vincent Vandeghinste for Dutch. The internship of the first author, during which this work has been done, was funded by the Lorraine Université d’Excellence OLKI research project\(^{13}\).

References


\(^{12}\)https://gitlab.inria.fr/ud-greenberg/ranlp-2021

\(^{13}\)https://olki.loria.fr/
<table>
<thead>
<tr>
<th>Language</th>
<th>Corpora</th>
<th>Without isubj</th>
<th>Ratio</th>
<th>With isubj</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech</td>
<td>CAC</td>
<td>SVO</td>
<td>4.27</td>
<td>SVO</td>
<td>5.28</td>
</tr>
<tr>
<td></td>
<td>CLTT</td>
<td>SVO</td>
<td>6.85</td>
<td>SVO</td>
<td>8.18</td>
</tr>
<tr>
<td></td>
<td>FicTree</td>
<td>NDO (SVO/SOV)</td>
<td>1.97</td>
<td>SVO</td>
<td>2.20</td>
</tr>
<tr>
<td></td>
<td>PDT</td>
<td>SVO</td>
<td>3.36</td>
<td>SVO</td>
<td>3.96</td>
</tr>
<tr>
<td></td>
<td>PUD</td>
<td>SVO</td>
<td>6.58</td>
<td>SVO</td>
<td>6.17</td>
</tr>
<tr>
<td>Estonian</td>
<td>EDT</td>
<td>SVO</td>
<td>3.80</td>
<td>NDO (SVO/SOV)</td>
<td>3.19</td>
</tr>
<tr>
<td></td>
<td>EWT</td>
<td>SVO</td>
<td>2.05</td>
<td>NDO (SVO/SOV)</td>
<td>1.70</td>
</tr>
<tr>
<td>German</td>
<td>GSD</td>
<td>NDO (SOV/SVO)</td>
<td>1.03</td>
<td>NDO (SOV/SOV)</td>
<td>1.03</td>
</tr>
<tr>
<td></td>
<td>HDT</td>
<td>NDO (SOV/SVO)</td>
<td>1.87</td>
<td>SOV</td>
<td>2.01</td>
</tr>
<tr>
<td></td>
<td>LIT</td>
<td>NDO (SOV/SVO)</td>
<td>2.30</td>
<td>SOV</td>
<td>2.53</td>
</tr>
<tr>
<td></td>
<td>PUD</td>
<td>NDO (SOV/SVO)</td>
<td>1.47</td>
<td>NDO (SOV/SOV)</td>
<td>1.62</td>
</tr>
<tr>
<td>Latin</td>
<td>ITTB</td>
<td>NDO (SOV/SOV)</td>
<td>1.22</td>
<td>NDO (SOV/SOV)</td>
<td>1.12</td>
</tr>
<tr>
<td></td>
<td>LLCT</td>
<td>NDO (OSV/SVO)</td>
<td>1.07</td>
<td>NDO (SOV/SOV)</td>
<td>1.40</td>
</tr>
<tr>
<td></td>
<td>PROIEL</td>
<td>NDO (SOV/SOV)</td>
<td>1.21</td>
<td>NDO (SOV/SOV)</td>
<td>1.16</td>
</tr>
<tr>
<td></td>
<td>Perseus</td>
<td>SOV</td>
<td>2.42</td>
<td>SOV</td>
<td>2.17</td>
</tr>
<tr>
<td>Turkish-German</td>
<td>SAGT</td>
<td>NDO (SOV/SOV)</td>
<td>1.95</td>
<td>SOV</td>
<td>2.22</td>
</tr>
</tbody>
</table>

Table 4: Corpora for which the word order changes with/without isubj and associated ratio (in bold, corpora for which the order changes).

Figure 6: Cosine values between the Latin corpora, with isubj on the left, without isubj on the right.


Kim Gerdes, Bruno Guillaume, Sylvain Kahane, and Guy Perrier. 2019a. Improving Surface-syntactic Universal Dependencies (SUD): surface-syntactic
<table>
<thead>
<tr>
<th>Corpora</th>
<th>first rank without isubj</th>
<th>first rank with isubj</th>
<th>diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>WELSH-CCG</td>
<td>VSO (80.6%)</td>
<td>VSO (71.1%)</td>
<td>-9.5%</td>
</tr>
<tr>
<td>LATIN-LLCT</td>
<td>OSV (32.0%)</td>
<td>SOV (42.7%)</td>
<td>+10.7%</td>
</tr>
<tr>
<td>AKKADIAN-RIAO</td>
<td>SOV (56.6%)</td>
<td>SOV (66.3%)</td>
<td>+9.7%</td>
</tr>
<tr>
<td>ICELANDIC-IcePAHC</td>
<td>SVO (57.0%)</td>
<td>SVO (63.4%)</td>
<td>+6.4%</td>
</tr>
<tr>
<td>WOLOF-WTB</td>
<td>SVO (59.3%)</td>
<td>SVO (64.8%)</td>
<td>+5.5%</td>
</tr>
<tr>
<td>OLD_CHURCH_SLAVONIC-PROIEL</td>
<td>SVO (52.0%)</td>
<td>SVO (57.4%)</td>
<td>+5.5%</td>
</tr>
<tr>
<td>CZECH-FicTree</td>
<td>SVO (47.6%)</td>
<td>SVO (52.7%)</td>
<td>+5.1%</td>
</tr>
</tbody>
</table>

Table 5: Corpora for which the first rank difference with or without isubj is greater than 5% (in bold, the corpus for which the first rank changes).


RED: A Novel Dataset for Romanian Emotion Detection from Tweets

Alexandra Ciobotaru
Advanced Technologies Institute
Bucharest, Romania
alexandra.ciobotaru@dcti.ro

Liviu P. Dinu
University of Bucharest
Bucharest, Romania
liviu.p.dinu@gmail.com

Abstract
In Romanian language there are some resources for automatic text comprehension, but for Emotion Detection, not lexicon-based, there are none. To cover this gap, we extracted data from Twitter and created the first dataset containing tweets annotated with five types of emotions: joy, fear, sadness, anger and neutral, with the intent of being used for opinion mining and analysis tasks. In this article we present some features of our novel dataset, and create a benchmark to achieve the first supervised machine learning model for automatic Emotion Detection in Romanian short texts. We investigate the performance of four classical machine learning models: Multinomial Naive Bayes, Logistic Regression, Support Vector Classification and Linear Support Vector Classification. We also investigate more modern approaches like fastText, which makes use of subword information. Lastly, we fine-tune the Romanian BERT for text classification and our experiments show that the BERT-based model has best performance for the task of Emotion Detection from Romanian tweets.

Keywords: Emotion Detection, Twitter, Romanian, Supervised Machine Learning

1 Introduction
Romanian language is a very little explored language in terms of Natural Language Processing (NLP) and Natural Language Understanding (NLU), but social media is full of data and people sharing their opinions about diverse topics, thus some opinion mining tools would be very useful.

Opinion mining is defined in Ravi (2015) as “the task of detecting, extracting and classifying opinions, sentiments and attitudes concerning different topics, as expressed in textual input”. In NLP tasks, there are two major subcategories of opinion mining: Sentiment Analysis (SA) and Emotion Detection (ED). The main difference between these two is that sentiment analysis categorizes opinions between positive or negative, with degrees of polarity, while emotion detection aims to extract the specific emotion a text gives to the reader.

English language is rich in NLP resources - we can see not only that there are many manually labelled datasets, with a large variety of emotions, but there also exist datasets with tweets labeled by emotion intensity (for example, WASSA dataset by Mohammad and Bravo-Marquez (2017) which was first presented at the 2017 Shared Task on Emotion Intensity). Detecting emotions can be a subjective task, even psychologists didn’t agree upon what the main emotions of a human being are. Plutchik (1973) identified eight primary emotions: ecstasy, admiration, terror, amazement, grief, loathing, rage and vigilance, while Ekman (2005) said that the primary emotions are: joy, sadness, fear, disgust, surprise and anger. Inspired by the work of Mohammad and Bravo-Marquez (2017), we decided to collect tweets from Twitter and manually annotate them for four emotions: fear, anger, joy, sadness, and also add a neutral class. We show the quality of the overall dataset by computing statistic descriptors and finally we train machine learning classifiers in order to create the first qualitative Romanian tool for emotion detection.

The usefulness of having a tool that detects emotions from texts varies from exploring customer opinions in order to ensure business growth, to improving human-computer interactions (HCI) in applications like onboarding chatbots and personal digital assistants (Strapparava and Mihalcea (2008)).

2 Recent Works
It has been shown by Acheampong et al. (2020) that text based ED research studies have been given less attention than other modes of ED, for instance
multimodal emotion detection including speech, body language, facial expressions, and so on.

There are two principal ways of assessing the ED problem: by using rule construction techniques to identify emotive words and word combinations, and machine learning approaches, which can be supervised or unsupervised. While unsupervised methods learn from unstructured data and could provide important clustering insights in opinion mining tasks, supervised methods give better detection rates (Canales and Martinez-Barco (2014)), but only if there exist some quality labeled data to give as input to the classifier. Combining these two main approaches gives a third way of detecting emotions from text - the hybrid approach, which takes the advantages of rule-construction and amplifies them with the power of machine learning. The latter was used in this paper.

Alotaibi (2019) used the ISEAR dataset to train a supervised model of emotion detection with Logistic Regression. The ISEAR (International Survey on Emotion Antecedents and Reactions) dataset was first introduced in the work of Scherer and Wallbott (1994) and contains 7666 emotional sentences labelled with 7 types of emotions: anger, disgust, fear, sadness, shame, joy and guilt. Abdel Razek and Frasson (2017) used the same dataset to test their dominant meaning approach in detecting emotions in chat messages.

For detecting emotions in tweets, Shah et al. (2019) proposed a hybrid approach consisting of lexical based approaches that use WordNet-Affect and EmoSenticNet with supervised classifiers trained on AIT-2018 dataset for English. This dataset was introduced in Semeval-2018 Task 1: Affect in Tweets by Mohammad et al. (2018) and consists of tweets annotated in 3 languages, for anger, fear, joy and sadness.

Ghanbari-Adv and Mosleh (2019) presented an ensemble classifier based on NLP techniques like Doc2Vec and 1500 k-Nearest Neighbor, Multilayer Perceptron and Decision Tree basic classifiers optimized with Parzen Tree Estimator (TPE), to detect emotions from ISEAR and OANC datasets, and also from an unstructured dataset of tweets from Crowdflower. Their results show an outstanding accuracy of 99.49% for regular sentences and 88.49% for irregular sentences.

Polignano et al. (2019) tried emotion detection in text using word-embeddings like Word2Vec, GloVe and FastText on their designed model, having as training datasets ISEAR, AIT-2018 and SemEval-2019 Task3 dataset. Their research concluded that FastText had slightly better performances.

Huang et al. (2019) investigated fine-tuning BERT (Bidirectional Encoder Representations from Transformers) on two datasets: EmotionLines with dialogues from Friends Sitcom, and Emotion-Push containing Facebook messenger chats. Their model obtained a micro f-score of 0.815 on Friends and 0.885 on EmotionPush. Acheampong et al. (2020) fine-tuned BERT with a Bi-LSTM classifier and obtain an average f-score of 0.73 on the ISEAR dataset. Chiorrini et al. (2021) also investigated the use of BERT for both sentiment analysis and emotion detection in Twitter data, using WASSA dataset (Mohammad and Bravo-Marquez (2017)), a shorter version of AIT-2018 containing 6755 tweets annotated for: sadness, fear, anger and happiness.

For more exotic languages, however, researchers had to manually annotate sentences or words in order to have a high quality dataset to work on. For Arabic, Almanie et al. (2018) developed a dataset of 4000 emotional words, including emojis, which they used to classify real-time tweets into 5 types of emotions (happy, sad, angry, scared, surprised). Grover and Verma (2016) used a hybrid approach to detect emotions from Punjabi texts - first they consider a rule-based engine to detect if the sentence has an emotion or not, and then they applied Support Vector Machine (SVM) and Naïve Bayes (NB) classifiers to detect 6 emotions: happy, fear, anger, sadness, disgust and surprise. Jayakrishnan et al. (2018) created a corpus of manually labelled Malayalam texts (an Indian dialect) into emotions like: sad, happy, anger or fear. Further they used a SVM classifier to create emotion detection models.

For Romanian language, there were some studies conducted on speech datasets with annotated emotions. For instance, Feraru and Zbancioc (2015) presented a method for emotion detection in Romanian speech that use Largest Lyapunov exponent of the Mel-frequency energy bands, with SVM and WKNN (Weighted K-Nearest Neighbors) classifiers, trained on the SRoL dataset (developed by Feraru et al. (2010)). Franti et al. (2017) created a deep learning model of Convolutional Neural Networks (CNN) trained on a set of recordings in Romanian language. Pavaloi et al. (2014) used three sets of recordings in Romanian language, annotated for positive and negative emotions, and trained
models using k-NN and SVM classifiers. In terms of text data, Lupea et al. (2021) present an unsupervised clustering approach used to mine emotional patterns in Mihai Eminescu’s poetry, based on the Romanian Emotion Lexicon created by Lupea and Briciu (2019) for feature extraction. In terms of sentiment analysis for Romanian language, Istrate and Ciobotaru (2021) created a dataset of tweets annotated for positive/negative sentiment and trained several classifiers on it, both classical and modern. There also exist some lexicon based approaches for Romanian sentiment analysis, like BabelSenticNet, a multilingual concept-level knowledge base described in Vilares et al. (2018).

Inspired by some of the works presented in this section, we created the first dataset of Romanian tweets labelled with five different emotions (anger, fear, sadness, happiness and neutral), in order to obtain a solid tool for detecting emotions in texts.

3 Data

In this section we present in detail our novel dataset, RED (Romanian Emotion Detection).

3.1 Scrapping Process

We considered the work of Mohammad and Bravo-Marquez (2017) where they explained their methodology of creating the first annotated datasets for fear, anger, joy and sadness, and we create a similar dataset, but for the Romanian language. In our work we construct the same four classes of emotions and also add a neutral class, as it has been previously shown by Al-Rubaiee et al. (2016) the importance of having a neutral class when classifying sentiments or emotions. This way, the classifier will not be forced to classify information he wasn’t trained to recognize, in one of the four classes created.

First, we create lists of query words correspondent to each of the class of emotions, which are:

- synonyms of the word defining the class - synonyms for “fear”, “anger”, “joy”, “sadness” and “neutral”, extracted from two synonym sources: an online dictionary1 and the work of Bulgar (1995), and
- jargon and commonly used words that express a certain feeling matching the class of emotion for which the scrapping process is conducted.

The total number of query words gathered for each class is detailed in Table 1. These main words are further expanded into their word families by adding prefixes and suffixes to their stem-words. Moreover, all morphological variants are generated for each resulted word. In this way, for each query-word a list or words is generated, and the resulting lists of words were further used for scraping tweets, using Snscrape2 python library.

Tweets were gathered from Twitter in the time-frame: 1st of February 2020 - 1st of February 2021, and checked for Romanian language using langdetect3 python library.

All gathered tweets were scraped from public accounts. Still, to protect confidentiality and anonymity of Twitter users, we removed usernames and also all proper nouns from tweets in the final dataset, using preprocessing techniques described in Section 3.3.

3.2 Annotation Process

The annotation process involved 3 annotators: Annotator 1 and Annotator 2, native Romanian speakers who decided over the same tweets, and Annotator 3, a psychologist, also native Romanian speaker, who tipped the scales regarding the tweets where Annotator 1 and Annotator 2 did not agree upon.

The result of the scraping process for each query word was a spreadsheet containing all the tweets found in the mentioned time-frame, having in their composition at least one word expanded from the query word. These spreadsheets were shuffled and manually checked by Annotator 1 if tweets indeed represented the emotion conveyed by the class they were scraped for. Maximum 50 tweets were kept for each query word, in order not to bias the classification process. An important rule for annotating was to annotate tweets that clearly expressed the researched emotion.

After the work of Annotator 1 was done, a number of approximately 1000 tweets resulted per class. In order the create a high quality dataset, these tweets were double-checked by Annotator 2 in order to make sure that the tweets indeed represent the emotion labelled by the first annotator. The two annotators disagreed upon 223 tweets for Anger, 251 for Fear, 309 for Joy, 279 for Sadness and 210 for Neutral class. These selected tweets were checked by Annotator 3, who gave his verdict regarding their conveyed emotion. Further, tweets

---

1https://www.sinonime.com/dex
2https://github.com/JustAnotherArchivist/snscrape
3https://pypi.org/project/langdetect/ version 1.0.8
were checked for duplicates and shuffled. The final number of remaining tweets is shown in Table 1. It can be observed that classes remain balanced after the quality check involved in the annotation process.

Table 1: Number of query words and final number of labelled tweets per class

<table>
<thead>
<tr>
<th>Class name</th>
<th>Query words</th>
<th>Labelled tweets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anger</td>
<td>35</td>
<td>807</td>
</tr>
<tr>
<td>Fear</td>
<td>25</td>
<td>778</td>
</tr>
<tr>
<td>Joy</td>
<td>32</td>
<td>876</td>
</tr>
<tr>
<td>Sadness</td>
<td>29</td>
<td>781</td>
</tr>
<tr>
<td>Neutral</td>
<td>24</td>
<td>805</td>
</tr>
</tbody>
</table>

3.3 Dataset Preprocessing

In order to have a high quality dataset, suitable for machine training, we performed some text preprocessing by removing unnecessary information from tweets that could potentially bias the classification:

- usernames in the form of @username;
- hiperlinks;
- hashtag sign (#), but the hashtag word was preserved, as it can contain relevant information for the classification problem;
- artefacts like &amp and \n;
- proper nouns, using Named Entity Recognition pipeline for Romanian from spacy\(^4\) (ro_core_news_sm).

The first four preprocessing steps were performed using regex. Emoticons, emojis, as well as all punctuation marks were left untouched by preprocessing techniques, as they convey emotions per se.

Final dataset was created by gathering all labelled tweets and shuffling them all. Further, the dataset was split into 3237 tweets for training, 405 tweets for validation and 405 tweets for testing, and this split was used for training all classifiers presented in Section 4.

3.4 Dataset Analysis

In Figure 1 we show the distribution of tokens per tweet for each class of emotions, in order to make sure all five classes have approximately the same distribution. As it can be seen, Fear, Joy and Neutral have approximately the same distributions, while Anger differs the most.

In Table 2 we compute descriptive statistics using R function aggregate. It can be seen that the longest tweet pertains to the Joy class, with 75 tokens, while the majority of tweets lie under 35 tokens, as the highest median is of 30 tokens, belonging to the Anger class. Mean and median are close to each other for this class, but for Sadness class, for instance, mean and median are very different, and we can see that for this class, the majority of tweets are shorter than for the other classes, meaning that Twitter users express sadness using less words in Romanian. Extrapolating, we can also state that Twitter users seemingly express anger in Romanian using more words than they would when expressing other emotions, like fear or joy.

Table 2: Descriptive statistics of token distributions in tweets.

<table>
<thead>
<tr>
<th>Emotion</th>
<th>Min</th>
<th>Median</th>
<th>Mean</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anger</td>
<td>2.00</td>
<td>30.00</td>
<td>31.97</td>
<td>68.00</td>
</tr>
<tr>
<td>Fear</td>
<td>2.00</td>
<td>21.00</td>
<td>24.18</td>
<td>70.00</td>
</tr>
<tr>
<td>Joy</td>
<td>2.00</td>
<td>21.00</td>
<td>25.27</td>
<td>75.00</td>
</tr>
<tr>
<td>Neutral</td>
<td>1.00</td>
<td>11.00</td>
<td>21.34</td>
<td>67.00</td>
</tr>
<tr>
<td>Sadness</td>
<td>1.00</td>
<td>16.00</td>
<td>28.36</td>
<td>74.00</td>
</tr>
</tbody>
</table>

Figure 1: Distribution of tokens per tweet for each class of emotions.

4 Models

In this section we describe the tried models in order to generate the first supervised emotion detector for Romanian short texts. First, we create a benchmark
composed of classical machine learning models, and second we try two more modern approaches, which make use of word embeddings: one that uses Facebook’s fastText word embeddings and a fine-tuned BERT classifier.

4.1 Classical Machine Learning Models

Classical machine learning models can sometimes have good results for modern problems. Thus, before diving into more modern solutions we first investigate Support Vector Classification (SVC), Linear Support Vector Classification (LinearSVC), Logistic Regression and Multinomial Naive Bayes (MultinomialNB) algorithms.

For feature extraction we use Term Frequency - Inverse Document Frequency (tf-idf) to represent tweets in vector form, which is a measure that tries to estimate the importance of tokens in the dataset by computing two statistics: term frequency (the number of appearances of a word in the whole dataset - see Eq.1) and inverse document frequency (the number of tweets in relation to the number of tweets containing the word - see Eq.2), as explained in Sammut (2010).

\[ tf = \frac{\text{No. Of Word Appear. in Tweet}}{\text{No. of Words in Tweet}} \]  
Eq.1

\[ idf = \frac{\text{No. Of Tweets}}{\text{No. of Tweets with Word}} \]  
Eq.2

The final result of the tf-idf is obtained by multiplying Eq.1 and Eq.2 (AlZoubi et al. (2020)):

\[ tf - idf = tf \times idf \]  
Eq.3

Practically, we convert tweets into a matrix of tf-idf features using TfidfVectorizer\(^5\), with the following characteristics:

- sublinear_df parameter is set to True in order to use a logarithmic form for term frequency, because it seems unlikely that twenty occurrences of a term in a document truly carry twenty times the significance of a single occurrence (Manning Christopher (2008)). Thus, Eq. 3 becomes:

\[ wtf - idft,d = wtf_{t,d} \times idft \]  
Eq.4

where:

\[ wtf_{t,d} = \begin{cases}  
\log tf_{t,d} & \text{if } tf_{t,d} > 0 \\
0 & \text{otherwise} 
\end{cases} \]  
Eq.5

- min_df parameter is set to 5, being the minimum number of tweets a word must be present in to be kept in the feature vector;
- ngram_range parameter is set to (1, 2) to indicate that we want to consider both unigrams and bigrams in our vector representation.

For label encoding we used Sklearn LabelEncoder\(^6\).

4.2 FastText Based Model

FastText\(^7\) is an open-source library, developed by Facebook AI Research lab with the purpose of text classification and representation. As Bojanowski et al. (2016) described in their work, fastText creates word representations based on the skipgram model, where each word is represented as a bag of character n-grams. A vector representation is associated to each character n-gram, words being represented as the sum of these representations. Using character level information helps capture the meaning of shorter words and allows the embeddings to map suffixes and prefixes.

The character n-gram selection is done using a sliding window between the minimum value of the character n-gram and the maximum value of the character n-gram. The word is stored in memory like the sum of character n-grams. For classification, word representations are averaged into a text representation to form a hidden variable, which is in turn fed to a linear classifier (Joulin et al. (2016)).

4.3 BERT Based Model

Another modern approach for text encoding is using pretrained vector representations. BERT (Bidirectional Encoder Representations from Transformers), introduced by Devlin et al. (2019) in their work, is considered state-of-the-art in many natural language processing tasks that use language representation. We fine-tune BERT model in order to obtain an emotion detection classifier.

BERT-base model contains an encoder with 12 Transformer blocks, 12 self-attention heads, and the hidden size of 768 (Chi et al. (2019)). To use a pre-trained BERT model, we first need to convert the input data into an appropriate format so that each sentence can be sent to the pre-trained model in order to obtain the corresponding embedding.


\(^7\)https://fasttext.cc/
For this task we use HuggingFace’s transformers package, and in particular the tokenizer for Romanian and the BERT pretrained model for Romanian described by Dumitrescu et al. (2020). The Romanian model was trained on three Romanian corporuses: OPUS, OSCAR and Wikipedia.

BERT tokenizer adds special tokens to the input text, converts all tokens into their corresponding IDs in the model, and adds the attention mask. The attention mask is a vector of 1 and 0 which tells the model which tokens should be taken into consideration and which should not. The resulted vectors are used to train the model.

5 Experiments and Results

In this section we present the results obtained on the test set for the previously described models.

5.1 Classical Machine Learning

The investigated algorithms are SVC, LinearSVC, Logistic Regression and Multinomial NB. For training we use sklearn off-the-shelf functions, without any other parameter modification. The accuracies we obtained for each model are shown in Table 3.

Table 3: Comparison between four classical machine learning models

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinearSVC</td>
<td>82.96%</td>
</tr>
<tr>
<td>LogisticRegression</td>
<td>78.77%</td>
</tr>
<tr>
<td>MultinomialNB</td>
<td>76.79%</td>
</tr>
<tr>
<td>SVC</td>
<td>76.30%</td>
</tr>
</tbody>
</table>

According to the results, LinearSVC has the best accuracy, 82.96%. It can be seen that the best model between the four models in Table 3 is LinearSVC, with a surprisingly high accuracy of almost 83%. Second comes Logistic Regression with an accuracy of 78.77%. For LinearSVC we compute the confusion matrix and the normalised confusion matrix and show results in Appendix, Figure 2. Normalization of the confusion matrix is useful in the case of not such perfectly balanced classes, to be able to visual interpret which class is being misclassified the most. It can be seen in the normalized confusion matrix in Appendix, Figure 2, that Sadness class is being misclassified most frequently, and the Neutral class is being classified correctly most often.

We also analyze precision, recall and f-score for each class, which are computed in Table 4, using classification_report method. In accordance with the confusion matrix, Neutral class has the highest precision, recall and F-score scores (not surprisingly, these values are also equal), while Sadness class has lowest scores.

Table 4: Classification report for Linear SVC

<table>
<thead>
<tr>
<th>Emotion</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joy</td>
<td>0.83</td>
<td>0.78</td>
<td>0.80</td>
<td>89</td>
</tr>
<tr>
<td>Fear</td>
<td>0.79</td>
<td>0.85</td>
<td>0.82</td>
<td>72</td>
</tr>
<tr>
<td>Anger</td>
<td>0.85</td>
<td>0.84</td>
<td>0.85</td>
<td>83</td>
</tr>
<tr>
<td>Neutral</td>
<td>0.89</td>
<td>0.89</td>
<td>0.89</td>
<td>81</td>
</tr>
<tr>
<td>Sadness</td>
<td>0.78</td>
<td>0.80</td>
<td>0.79</td>
<td>80</td>
</tr>
<tr>
<td>Macro avg</td>
<td>0.82931</td>
<td>0.83095</td>
<td>0.82972</td>
<td>405</td>
</tr>
</tbody>
</table>

Combining the results from Appendix, Figure 2 and Table 4 we see that out of 80 tweets labelled with Sadness, the model correctly classified only 64, 8 being classified as Joy, and out of 89 tweets labelled with Joy, the model correctly classified only 69, 7 being classified as Sadness. This means that the model confuses Joy with Sadness and vice-versa, which is counter-intuitive.

5.2 FastText Based Model

To train the fastText based model we used the train_supervised method from fasttext library, with the word n-grams parameter set to 2, because for the classical machine learning models considered previously we have also used word n-grams of 2 during encoding. We conducted a set of four experiments for the fastText based model, two experiments using pretrained vectors for the Romanian language, and another two experiments where we let the model autotune on the validation set, without taking into consideration pretrained word vectors. Results are shown in Table 5, along with the hyperparameters used for each model, which are:

- \( \text{lr} \) - learning rate;
- \( \text{vectors} \) - fastText pretrained vectors for Romanian were trained on Common Crawl and Wikipedia using CBOW with position-weights, in dimension 300, with character n-grams of length 5, a window of size 5 and 10 negatives;
- \( \text{ws} \) - size of the context window;
- \( \text{wNgrams} \) - maximal length of word n-gram;
- \( \text{epoch} \) - number of epochs used for training.

Table 5: FastText Based Model

Combining the results from Appendix, Figure 2 and Table 4 we see that out of 80 tweets labelled with Sadness, the model correctly classified only 64, 8 being classified as Joy, and out of 89 tweets labelled with Joy, the model correctly classified only 69, 7 being classified as Sadness. This means that the model confuses Joy with Sadness and vice-versa, which is counter-intuitive.
• autotune - time set for autotuning on the validation file;
• minm - minimum length of character n-gram;
• maxn - maximum length of character n-gram.

As seen in Table 5, models trained without making use of pretrained vectors for Romanian had better performances than the models trained with these vectors, but this can be due to the fact that experiments without pretrained vectors were done using hyperparameter autotuning on the validation set, and might not be necessarily related to using pretrained vectors or not. At the time these experiments were conducted, train_supervised method from fastText library didn’t allow autotuning of hyperparameters using pretrained vectors.

Table 5: Hyperparameters and performance of tried fastText models

<table>
<thead>
<tr>
<th>Model</th>
<th>lr</th>
<th>vectors</th>
<th>ws</th>
<th>wNgrams</th>
<th>epoch</th>
<th>minm</th>
<th>maxn</th>
<th>tune</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model1</td>
<td>0.001</td>
<td>yes</td>
<td>2</td>
<td>2</td>
<td>1000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.733</td>
</tr>
<tr>
<td>Model2</td>
<td>0.001</td>
<td>yes</td>
<td>2</td>
<td>2</td>
<td>1000</td>
<td>2</td>
<td>3</td>
<td>-</td>
<td>0.795</td>
</tr>
<tr>
<td>Model3</td>
<td>-</td>
<td>no</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>3</td>
<td>600</td>
<td>0.795</td>
<td></td>
</tr>
<tr>
<td>Model4</td>
<td>-</td>
<td>no</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>600</td>
<td>0.847</td>
<td></td>
</tr>
</tbody>
</table>

For Model 4 we compute the confusion matrix (Appendix, Figure 3), both classic and normalized. We can see from the normalized confusion matrix that the model has an overall best performance on the Anger class, and worst performance on the Neutral class. But if we examine performance in detail, we see in the classification report (Table 6) that although Anger class has the highest recall, it lacks precision, and while the Neutral class has high precision, it lacks recall. Also, analyzing the confusion matrix we observe many high values of missclassification, for each class: from 89 tweets labelled Joy the model classified 6 as Sadness; from 72 tweets labelled Fear, the model classified 7 as Anger; from 83 tweets labelled as Anger, the model classified 6 as Sadness; from 81 tweets labelled Neutral the model classified 8 as Joy, and from 80 tweets labelled Sadness the model classified 6 as Anger. We can conclude that the fastText based model may have an overall good performance taking into account all classes at once, but behaves poorly in discrete mode.

5.3 BERT Based Model

To train the BERT model we preprocessed data as explained in Section 4.3, using the tokenizer.encode_plus method from Huggingface. We fine-tuned BERT model by adding a classifier for the task of emotion detection, comprised of:

- a drop-out layer with probability 0.3. As explained in Hinton et al. (2012), drop-out layers are used for regularization and preventing the co-adaptation of neurons;
- a fully connected layer that applies a linear transformation to data;
- a transformation of the output using Softmax function.

Training was done using Cross Entropy loss function with an AdamW optimizer for the learning rate. Although BERT authors have some recommendations in their paper, Devlin et al. (2019), we opted for a batch size of 8 and 5 epochs, because our 16GB GPU card couldn’t handle a bigger batch size, and in 5 epochs the model already reached its highest accuracy. The maximum utterance length was set to 100.

For fine-tuning BERT we chose Cross Entropy loss function and Softmax optimizer, as these are usually used with multiclass classification problems.

The confusion matrices, both simple and normalized, are shown in Appendix, Figure 4. It can be observed on the normalized confusion matrix that Sadness class has been misclassified most often, while Joy and Fear have best accuracies per class. On the simple confusion matrix we can see that misclassifications don’t have such high values like on the other tried models’ confusion matrices, none exceeding 5 wrong classifications.

The overall accuracy for the BERT based model is 90.37%, fact that can also be observed in the confusion matrix, by summing up the values on the diagonal and dividing to the whole number of samples (405 tweets in the test set).

This model’s classification report is presented in Table 7. The macro averaged precision, recall and F-score are very similar, and to the overall accuracy of the model as well. Best precision, recall and F-score belong to the Joy class, worst precision and F-score to Anger class, and worst recall to Sadness class.

Table 6: Classification report for fastText Model 4

<table>
<thead>
<tr>
<th>Emotion</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joy</td>
<td>0.89</td>
<td>0.85</td>
<td>0.87</td>
<td>89</td>
</tr>
<tr>
<td>Fear</td>
<td>0.88</td>
<td>0.85</td>
<td>0.87</td>
<td>72</td>
</tr>
<tr>
<td>Anger</td>
<td>0.78</td>
<td>0.88</td>
<td>0.82</td>
<td>83</td>
</tr>
<tr>
<td>Neutral</td>
<td>0.90</td>
<td>0.79</td>
<td>0.84</td>
<td>81</td>
</tr>
<tr>
<td>Sadness</td>
<td>0.80</td>
<td>0.86</td>
<td>0.83</td>
<td>80</td>
</tr>
<tr>
<td>Macro avg</td>
<td>0.85170</td>
<td>0.84666</td>
<td>0.84742</td>
<td>405</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>lr</th>
<th>vectors</th>
<th>ws</th>
<th>wNgrams</th>
<th>epoch</th>
<th>minm</th>
<th>maxn</th>
<th>tune</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model1</td>
<td>0.001</td>
<td>yes</td>
<td>2</td>
<td>2</td>
<td>1000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.733</td>
</tr>
<tr>
<td>Model2</td>
<td>0.001</td>
<td>yes</td>
<td>2</td>
<td>2</td>
<td>1000</td>
<td>2</td>
<td>3</td>
<td>-</td>
<td>0.795</td>
</tr>
<tr>
<td>Model3</td>
<td>-</td>
<td>no</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>3</td>
<td>600</td>
<td>0.795</td>
<td></td>
</tr>
<tr>
<td>Model4</td>
<td>-</td>
<td>no</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>600</td>
<td>0.847</td>
<td></td>
</tr>
</tbody>
</table>
Table 7: Classification report for BERT based model

<table>
<thead>
<tr>
<th>Emotion</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joy</td>
<td>0.95</td>
<td>0.93</td>
<td>0.94</td>
<td>89</td>
</tr>
<tr>
<td>Fear</td>
<td>0.93</td>
<td>0.89</td>
<td>0.91</td>
<td>72</td>
</tr>
<tr>
<td>Anger</td>
<td>0.85</td>
<td>0.89</td>
<td>0.87</td>
<td>83</td>
</tr>
<tr>
<td>Neutral</td>
<td>0.87</td>
<td>0.91</td>
<td>0.89</td>
<td>81</td>
</tr>
<tr>
<td>Sadness</td>
<td>0.90</td>
<td>0.86</td>
<td>0.88</td>
<td>80</td>
</tr>
<tr>
<td>Macro avg</td>
<td>0.90437</td>
<td>0.90354</td>
<td>0.90366</td>
<td>405</td>
</tr>
</tbody>
</table>

5.4 Comparisons

Retrospectively, although LinearSVC had a high overall accuracy, it confused Joy and Sadness, but had good performance on the Neutral class. On the other hand, the fastText based model had worst performances on the Neutral class, and performed best when classifying Anger, but had an overall better accuracy than LinearSVC. Lastly, our BERT-based model outperforms all the other models considered, as seen in Table 8, where we aggregate all our results and compare models using accuracy, macro averaged precision, recall and F-score. We can see that the BERT-based model came out best regarding all measures, fastText based model came out second, and classical LinearSVC on the last place. A probable explanation for such good results is that pre-trained BERT learned contextual relations between words and fine-tuning the model makes use of these relations when classifying.

Table 8: Comparison between created ED models

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>90.37%</td>
<td>90.44%</td>
<td>90.33%</td>
<td>90.37%</td>
</tr>
<tr>
<td>fastText</td>
<td>84.70%</td>
<td>85.17%</td>
<td>84.67%</td>
<td>84.74%</td>
</tr>
<tr>
<td>LinearSVC</td>
<td>82.90%</td>
<td>82.93%</td>
<td>83.10%</td>
<td>82.97%</td>
</tr>
</tbody>
</table>

6 Conclusions and Future Works

In this article, we presented our novel dataset for emotion detection, the first dataset of its kind for the Romanian language. We researched the state-of-the-art and created a benchmark of machine learning models in order to obtain an automatic emotion detector from tweets, having the purpose of being used in real-life tasks, adjacent to the field of opinion mining.

Although the dataset is not very large, it provided enough data to generate a text emotion detection model with an accuracy of 90.37%. In the future, we plan to enlarge this dataset with more tweets per class, an action that most probably will increase accuracy of the models. Also, we plan on adding more classes of emotions, to generate an even more fine-grained dataset for detecting emotions in Romanian content.

7 Acknowledgments

We would like to thank Nicu Ciobotaru and Ioana Alexandra Răducanu for their help with the annotation process, Ligia Maria Bătrîncu for proof reading and suggestions, as well as the anonymous reviewers for their time and valuable comments.

We acknowledge the support of a grant of the Romanian Ministry of Education and Research, CCCDI—UEFISCDI, project number 411PED/2020, code PN-III-P2-2.1-PED-2019-2271, within PNCDI III.

References


Omar AlZoubi, Saja Al-Tawalbeh, and Mohammad AL-Smadi. 2020. Affect detection from arabic tweets using ensemble and deep learning techniques. *Journal of King Saud University - Computer and Information Sciences*.


Carlo Strapparava and Rada Mihalcea. 2008. Learning to identify emotions in text. pages 1556–1560. ACM.

Appendix

Figure 2: Confusion matrix for linear SVC model.

Figure 3: Confusion matrix for fastText based model.

Figure 4: Confusion matrix for BERT based model.
Assessing the Eligibility of Backtranslated Samples Based on Semantic Similarity for the Paraphrase Identification Task

Jean-Philippe Corbeil and Hadi Abdi Ghavidel
Polytechnique Montreal
{jean-philippe.corbeil,hadi.abdi-ghavidel}@polymtl.ca

Abstract

In the domain of natural language augmentation, the eligibility of generated samples remains not well understood. To gather insights around this eligibility issue, we apply a transformer-based similarity calculation within the BET framework based on backtranslation, in the context of automated paraphrase detection. While providing a rigorous statistical foundation to BET, we push their results by analyzing statistically the impacts of the level of qualification, and several sample sizes. We conducted a vast amount of experiments on the MRPC corpus using six pre-trained models: BERT, XLNet, Albert, RoBERTa, Electra, and DeBerta. We show that our method improves significantly these “base” models while using only a fraction of the corpus. Our results suggest that using some of those smaller pre-trained models, namely RoBERTa base and Electra base, helps us reach F1 scores very close to their large counterparts, as reported on the GLUE benchmark. On top of acting as a regularizer, the proposed method is efficient in dealing with data scarcity with improvements of around 3% in F1 score for most pre-trained models, and more than 7.5% in the case of Electra.

1 Introduction

Natural language processing (NLP) tasks require sufficiently large datasets to achieve the maximum robustness of the trained models. Low sizes of data pose the risks of hindering the models’ convergence during the training process, which leads to less accurate predictions (e.g. classification) or generations (e.g. translations). On the other hand, the provision of high-quality labelled data is often very expensive both in terms of money and time. As a result, NLP scientists seek alternative methods to tackle this issue. One of the solutions is the application of data augmentation techniques. These methods help considerably to alleviate insufficiencies regarding the quantity of labelled data and the expertise to annotate the data. These augmentation techniques are also proved to induce a regularization effect during the training of NLP models to avoid overfitting, most notably, on surface cues.

In this paper, we examine the impact of adding a post-processing stage after applying such data augmentation technique to assess the eligibility of the generated samples. We intend to run our analysis in automated paraphrase identification. In this regard, we increase the size of the paraphrase data through a backtranslation method called BET (Corbeil and Abdi Ghavidel, 2020). In particular, we conduct the following experiments:

- We take randomly several samples of the original train set and augment them with back-translation.
- After augmenting the textual data, we assess the eligibility by applying a similarity filter. We report the results for three criteria: 0.8, 0.9, and 0.95.
- We examine six pre-trained transformer models: BERT, XLNet, RoBERTa, ALBERT, Electra, and DeBerta.
- We run ten times each experiment randomizing the random seed to measure the averaged metrics and their p-values.

The remainder of this paper is structured as follows. In section 2, we describe the previous works in natural language augmentation. In section 3, we explain our methodology in terms of the dataset and our overall pipeline. Next, we illustrate and discuss the results. Finally, we summarize our findings and talk about the possible future research avenues.
Figure 1: Experimentation pipeline schema to generate the training set from a sampled version of the MRPC — $N$ randomly selected samples for each experiment — on top of which we add the eligible backtranslated examples. Using sentence-transformers $\vec{e}$ to encode the utterances as vectors, we estimate the qualification given a threshold $T$ by measuring the cosine similarity $\text{sim}$ between the generated sentences and the original ones. The utterance named Sentence 2 corresponds to the column name inside the MRPC dataset, which is identified as the paraphrase.

2 Related Work

Data augmentation has been intensively explored in computer vision given its straightforward geometrical nature, especially image processing. According to Feng et al. (2021), the popular techniques in this field are cropping, flipping, and colour jittering. From a natural language processing standpoint, many authors noted that the natural language augmentation methods (NLA) either attempt to preserve the meaning and structure after the data augmentation process (Corbeil and Abdi Ghavidel, 2020; Tong et al., 2019; Coulombe, 2018; Sennrich et al., 2016; Anaby-Tavor et al., 2019; Radford et al., 2019) or to modify the tokens without taking into account the overall structure of the language (Wei and Zou, 2019; Coulombe, 2018). Feng et al. (2021) classified the techniques into the following categories:

- **Rule-Based techniques**: In these techniques, the original examples are changed (rewritten) based on a set of pre-defined rules. For instance, Wei and Zou (2019) applied random insertion, deletion, and swap on the tokens of the sentences.

- **Example interpolation techniques**: These techniques, also called mixed sample data augmentation, either interpolate the feature vectors (Zhang et al., 2017) or fuse the original examples into pairs (Ghiasi et al., 2020).

- **Model-Based techniques**: This set of techniques are concentrated on training models to generate diverse examples out of the original counterparts. Paraphrase generation (Sennrich et al., 2016) is a widely-known example of such techniques.

To the best of our knowledge, none of the papers in the aforementioned categories has analyzed the effect of a post-processing stage so far. Only Coulombe (2018) and Corbeil and Abdi Ghavidel (2020) highlighted the necessity for filtering out the backtranslation outputs to assess the data augmentation validity, without conducting any specific experiment to support the claim.

In the current paper, we closely set our work on the BET framework proposed by (Corbeil and Abdi Ghavidel, 2020), on top of which we enrich the meaning preserving aspect with a semantic similarity stage. Their original approach uses a model-based technique by applying backtranslation on ten intermediary languages to obtain a soft data augmentation. Thus, they generate ten times the amount of original data. Then, they analyzed the
resulting improvements on the paraphrase detection task as external validation. They tested various pre-trained models: BERT (Vaswani et al., 2017), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019) and ALBERT (Lan et al., 2019). In this work, we carry out the experiments with the same pre-trained models adding the most recent ones: Electra (Clark et al., 2020) and DeBerta (He et al., 2020). There are still other closely related works such as (Shakeel et al., 2020), but the authors used neural network architectures such as LSTMs and CNNs through exploiting a set of hand-crafted features on the MRPC, Quora and SemEval datasets.

3 Methodology

3.1 Dataset

In this paper, we focus our experiments on the MRPC\(^1\) corpus. This paraphrase corpus is included in the GLUE benchmark (Wang et al., 2019). It consists of a pair of sentences (sentence and paraphrase), which are pulled from online news sources. Overall, 4076 pairs were allocated to the train set and 1725 to the test set. We further split the MRPC train set into a smaller train set (90%, 3,668 pairs) and a validation set (10%, 408 pairs).

3.2 Data Augmentation Pipeline

As illustrated in Figure 1, our pipeline includes a backtranslation process based on BET using the Google Translate API and a filtering process using the sentence-transformers bi-encoder approach (Reimers and Gurevych, 2019).

On the basis of BET, we selected ten languages for the backtranslation procedure. These intermediary languages are: Chinese (zh), Spanish (es), Arabic (ar), Japanese (ja), Telugu (te), Javanese (jv), Korean (ko), Vietnamese (vi), Turkish (tr) and Yoruba (yo). In this regard, we augment only the paraphrases (e.g. the column Sentence 2) through backtranslating them into English from one of the aforementioned languages.

Our filtering module is mainly based on the sentence-transformers bi-encoder approach (Reimers and Gurevych, 2019). It is built to compute a unique sentence representation by pooling all the transformer’s contextual word embeddings — applying the mean. It is optimized under cosine loss in a Siamese neural network fashion. We choose the stsb-distilroberta-v2 model, which is a lightweight version. Formally, we note it as a function \(\vec{e}(s)\) with \(s\) being the sentence to encode into a sentence embedding. Then, we calculate the cosine similarity (see equation 1) between the original sample and the backtranslated one. Finally, we opt-out the ones which are below various thresholds \(T \in \{0.95, 0.9, 0.8\}\).

\[
sim(s_1, s_2) = \frac{\vec{e}(s_1) \cdot \vec{e}(s_2)}{||\vec{e}(s_1)|| \cdot ||\vec{e}(s_2)||} \tag{1}
\]

We show that different thresholds \(T\) influence drastically the outcome of our transformer-based paraphrase identifiers. We can approximate the effect of the semantic filtering as a paraphrase verification \(para(\cdot, \cdot)\) like in equation 2. We hypothesize that, by adding this filtering stage, we can reinforce the preservation of meaning into BET up to some specific threshold \(T\).

\[
para(s_1, s_2) \approx \begin{cases} 
1 & \text{if } \sim(s_1, s_2) \geq T \\
0 & \text{else}
\end{cases} \tag{2}
\]

3.3 Adjusting the Thresholds for Understanding the Similarities

In Figure 2, we present the histograms representing the distributions of similarities between the original sentence and the backtranslated ones. We displayed one histogram for each intermediary language. Giving the proximity of our setup with the original BET setup, we observe that the amount of generated examples with a similarity above 0.95 — as it is sorted in Figure 2 — correlates with the results reported by original BET experiments (Corbeil and Abdi Ghavidel, 2020). For instance, the authors mentioned that Spanish (es) and Vietnamese (vi) are among the best intermediary languages to use with BET to achieve the most gain on the performances. From our observations, we conclude that looking at similarity is a better way to analyze the impact of intermediary languages on backtranslation.

Based on those distributions, we also set the three similarity thresholds used in our experiments. We selected 0.8 because it conserves a majority of the generated data while filtering outliers. Afterwards, we chose 0.9 which is a compromise between quantity and quality. Finally, 0.95 is the strictest threshold keeping only the most similar examples. We won’t extend our analysis to a threshold of 0 — equivalent to the original BET — since

\(^1\)Microsoft Research Paraphrase Corpus
0.8 encompasses most of the data and the rest should be only outliers.

Considering the full MRPC corpus, we further analyzed the total amount of eligible samples after applying the different similarity thresholds $T$ in the bar chart of Figure 3. We can see that the threshold of 0.8 retains most of the generated data. For a threshold of 0.9, a majority of samples are still qualified for the training of the model. The 0.95 threshold drops less than two-thirds of the data taking only the most similar examples to the original sentence. By observing the results of the experiments in the next section, we can conclude about which of the quantity criterion ($T = 0.8$) or the quality criterion ($T = 0.95$) is better to determine the eligibility of a backtranslated text. We externally assessed this qualification by measuring the performances achieved by the models on the paraphrase detection task.

4 Results and Discussion

As we mentioned in section 1, we evaluated our natural language augmentation approach on BERT (Devlin et al., 2018), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019), Electra (Clark et al., 2020) and DeBerta (He et al., 2020). We performed our experiments iteratively beginning with the sampling of only 100 random examples from the original MRPC dataset up to reaching the whole trainset. We selected ten sample sizes to cover low-data regime situations (100 - 1,000) and high-data regime situations (1,000 - 3,668). We leveraged the HuggingFace\textsuperscript{2} transformers library and the sentence-transformers library for all our fine-tuning and filtering experiments. We fixed the training configuration to well-known hyperparameters for this task based on HuggingFace’s recommendations. We left a granular optimization of the hyperparameters for future works. Thus, our experimental setup is as follows:

- Batch size: 32
- Learning rate: 3e-5
- Number of runs per experiment (random seeds were randomized at each run): 10
- Number of different experiments: 240
- Sample sizes: [100, 250, 500, 750, 1000, 1500, 2000, 2500, 3000, 3668]

\textsuperscript{2}https://huggingface.co/
• FP16 mode

Overall, we conduct 240 unique experiments. For each of these, we report the average of 10 runs in Figures 4, 7, and 8. Our evaluation metrics are respectively: F-1 score, precision, and recall. We mainly focus on the average of F-1 scores since it is the metric used for the GLUE benchmark and in the literature. Nonetheless, we also inspect the precision and recall — the components of the harmonic mean used to compute the F-1 score — to gain a thorough understanding of our method.

In Figure 4, we illustrate the F-1 scores for all six models across all the sampling sizes. We display four curves: baseline (plain MRPC without BET), BET filtered 0.95, BET filtered 0.9, and BET filtered 0.8. We also added the F1 scores (dashed black lines) reported by their original authors for the corresponding large models. As a first general observation, we observe that all the baseline curves have approximately an S-shaped trend, in which sharp variations occur. In contrast, the BET filtered lines are smooth logarithmic-like growth, mostly all above the baseline curve. We further note that the higher we fix the similarity threshold, the bigger the gains we have. Some models like RoBERTa and DeBERTa have gained between 0.04 and 0.08 in the sample size region between 500 and 1,500 samples.

We directly provided in Figure 6 the F-1 scores gain $G$ in percent computed by comparing the BET filtered 0.95 against the baseline. We used the equation

$$G = 100 \cdot \frac{F_1^{\text{augmented}} - F_1^{\text{baseline}}}{F_1^{\text{baseline}}}$$

(3)

Figure 6: F-1 score gain in percent by comparing the F1-score averages between the BET filtered 0.95 against the Baseline.

We also checked the p-values from the Student's T-test between all the augmented F-1 scores and the baseline ones, in Figure 5. In statistics, we are usually advised for a minimum of about 35 runs to benefit from the law of large numbers. Given the long training times of transformer models and the number of configurations set by our methodology, we limited our experiments to ten runs. However, we note it is already twice the usual five runs used in the literature with these models. The resulting comparison using the p-values is therefore limited. Yet, we observe that in the case of the BET filtered 0.95 mostly all F-1 scores are strongly significant below a p-value of 0.05 (dashed black line). However, we note less significant results at 100 samples, and some at high sample sizes. Those regions, as well as the BET filtered 0.8 and BET filtered 0.9, would require further runs to conclude statistically the T-test. We finally highlight that, in the case of RoBERTa and Electra in the large sample size region (from 3,000 to 3,668), the results reaching near their large counterparts are significant.

To have an idea of the underlying influences behind the reported F-1 scores, we also provide the values for precision and recall in Figures 7 and 8.

In Figure 7, we observe in many cases a reduction in precision. However, in the low sample sizes — below 1,500 samples —, we notice gains in precision between 0.03 and 0.15. Furthermore, we report that the lower the similarity filter is (0.8 and 0.9), the more we tend to degrade the precision compared to the baseline. We remark that all the BET filtered 0.95 curves are surpassing the baseline precision. We mentioned as a first hypothesis that a higher threshold on the similarity scores would
induce a higher quality of the generated samples — leading logically to a rise in precision. Therefore, we confirm the validity of this hypothesis based on its impact on the precision curves.

In Figure 8, we show the sensitivity curves, on which we denote two observations. First, as expected generally with data augmentation in NLP, we note an overall gain in recall when applying BET from a couple of percent up to 0.05. We observe that this gain tends to lower as the similarity threshold gets higher, but remains above the baseline. The pre-trained models that benefit the most in terms of sensitivity are respectively BERT, DeBerta, Albert and RoBERTa. XLNet and Electra obtained very low improvements on the sensitivity metric. When looking below a 1,000 sample sizes, we notice a drastic drop in recall from the baseline to any of the BET curves. Nonetheless, we rationalize that the models tend to declare a paraphrase too often. We conclude that this issue is solved by applying any backtranslation.

Figure 4: F-1 score curves for all experiments, where each point is the average of ten experiments. The dashed black lines are the GLUE benchmark scores reported for the large models. The model names are the ones used by the HuggingFace Model Hub.

Figure 5: P-values of the F1-score curves augmented by the BET framework against the Baseline curve.
5 Conclusion and Future Work

In this paper, we described a method based on back-translation which is followed by a filtering stage to keep the most eligible examples. We increased the F-1 scores on the automatic paraphrase detection task by up to 7.6% compared to the baseline using only a fraction of the original dataset. Furthermore, we demonstrated that this approach limits the gain in recall while avoiding degrading the precision, which results in the best F-1 scores. With the augmentation of the full dataset using RoBERTa base and Electra base, we achieved results that are close to the reported GLUE benchmark scores, while the original authors were using their corresponding large versions. In conclusion, pre-trained transformer models have very good transfer-learning capabilities, but they still largely benefit from the support of high-quality natural language augmentation, both to enrich very small datasets and to alleviate the overfit on surface cues.

In future work, we will extend this work to the other paraphrase corpus as well as to the other NLP tasks such as multi-class classification.
References


Fine-tuning Neural Language Models for Multidimensional Opinion Mining of English-Maltese Social Data

Keith Cortis
ADAPT Centre,
Dublin City University,
Glasnevin, Dublin 9, Ireland
keith.cortis
@adaptcentre.ie

Kanishk Verma
ADAPT Centre,
Dublin City University,
Glasnevin, Dublin 9, Ireland
kanishk.verma
@adaptcentre.ie

Brian Davis
ADAPT Centre,
Dublin City University,
Glasnevin, Dublin 9, Ireland
brian.davis
@adaptcentre.ie

Abstract
This paper presents multidimensional Social Opinion Mining on user-generated content gathered from newswires and social networking services in three different languages: English—a high-resourced language, Maltese—a low-resourced language, and Maltese-English—a code-switched language. Multiple fine-tuned neural classification language models which cater for i) English, Maltese and Maltese-English languages as well as ii) five different social opinion dimensions, namely subjectivity, sentiment polarity, emotion, irony and sarcasm, are presented. Results per classification model for each social opinion dimension are discussed.

1 Introduction
Social Opinion Mining on data obtained from social sources is an evolving research domain tasked with the identification of several opinion dimensions, such as subjectivity, sentiment polarity, emotion, irony and sarcasm, from noisy user-generated social data spread across heterogeneous sources (Cortis and Davis, 2021b). Currently, Social Opinion Mining is used in several real-world scenarios, namely chatbots (Androutsopoulou et al., 2019), adaptive customer online service based on identified customer sentiment and emotion (Yadollahi et al., 2017), tracking of overall customer satisfaction for a product or service (Zhao et al., 2019), and detection of changes in customer opinion towards a brand, product or service (Geetha et al., 2017).

This paper presents multidimensional Social Opinion Mining on user-generated content gathered from newswires and social networking services in three different languages: English—a high-resourced language, Maltese—a low-resourced language, and Maltese-English—a code-switched language. Our aim is use these initial results to improve cross-lingual performance of English-Maltese neural language models. Research applications of the developed classification models include opinion summarisation and fine-grained opinionated search of each dimension. This work is in line with Malta’s Strategy and Vision for Artificial Intelligence (Parliamentary Secretariat for Financial Services and Innovation, 2019), with current investment being made in the development of Maltese language resources and tools to counter the threat of “digital extinction” for the Maltese language, which has low technological support available in comparison with other European languages (Rosner et al., 2012).

We leverage a novel multidimensional and multilingual social opinion dataset in the socio-economic domain, specifically Malta’s annual Government Budget, which comprises social data from the 2018, 2019 and 2020 budgets to fine-tune pre-trained neural language models for benchmarking purposes.

2 Related Work
Nguyen et al. (Nguyen et al., 2020) developed the first large-scale pre-trained language model BERTweet for English tweets, which outperforms its baselines. Experiments were conducted on three NLP tasks, namely Part-of-Speech tagging, Named Entity Recognition and text classification, namely sentiment analysis and irony detection. For the latter task, the authors used the 3-class sentiment analysis dataset from SemEval-2017 Task 4A (Rosenthal et al., 2017) and the 2-class irony detection dataset from the SemEval-2018 Task 3A (Van Hee et al., 2018). The authors in (Croce et al., 2020) propose GAN-BERT which extends the fine-tuning of architectures similar to Bidirectional Encoder Representations from Transformers (BERT) (Devlin
et al., 2018), using unlabelled data in a generative adversarial setting. Experimental results show that around 50-100 annotated examples can still produce good performance in sentence classification tasks. Results are confirmed for sentiment analysis over the SST-5 dataset (Socher et al., 2013) containing 5-class sentiment polarity categories. Babanejad et al. (Babanejad et al., 2020) propose two novel deep neural network models for sarcasm detection by including affective and contextual features in the extended BERT architecture.

Certain studies focused on low-resourced languages, with (Fei and Li, 2020) investigating cross-lingual sentiment classification where the low-resource language does not have any labels or parallel corpus, (Grießhaber et al., 2020) exploring the reduction of trainable model parameters for fine-tuning a model with a small amount of data, (Koto et al., 2020) releasing a new pre-trained language model for Indonesian which was evaluated on several tasks such as sentiment analysis, and (Yimam et al., 2020) using RoBERTa (Liu et al., 2019)–a replication of BERT developed by Facebook– for exploring Amharic sentiment analysis from social media text.

Demszky et al. (Demszky et al., 2020) conduct transfer learning experiments on existing emotion benchmarks to show that the GoEmotions dataset of fine-grained emotions generalises across domains and taxonomies. The authors demonstrate that if little target domain labelled data is available, this dataset can be used as a baseline for emotion understanding. Similarly, the XED multilingual dataset for emotion detection catering for a total of 32 languages has been evaluated using language-specific BERT models (Öhman et al., 2020). Lastly, (Makarenkov and Rokach, 2020) explore several off-the-shelf BERT models, where they show that the complexity and computational cost of BERT does not provide a guarantee for an improved predictive performance for classification tasks. This is especially relevant in cases where small domain-specific datasets are used, which datasets are also imbalanced due to the minority class being under-represented.

3 Dataset

The dataset of multidimensional and multilingual social opinions for Malta’s Annual Government Budget² (Cortis and Davis, 2021a) is used for the work carried out in this paper. This dataset contains 6,387 online posts for the 2018, 2019, and 2020 budgets, which user-generated content was collected from newswires and social networking services. In terms of languages, the majority of the online posts were in English (74.09%), Maltese or Maltese-English (24.99%). Each online post is annotated for the following five social opinion dimensions: subjectivity, sentiment polarity, emotion, sarcasm and irony. Table 1 presents the overall class distribution of online posts for each social opinion dimension and the language annotation. Statistics are provided for the entire dataset (columns 2 and 3), the subset of online posts in English (columns 4 and 5) and subset of online posts in Maltese and Maltese-English (columns 6 and 7).

4 Experiments

All experiments have been carried out on Google Colaboratory³ using a Tesla K80/Tesla T4/Tesla P100-PCIE-16GB Graphics Processing Unit (GPU).

The baseline models experiments have been carried out in the Python programming language using Jupyter Notebook⁴ on a machine with an Intel(R) Core(TM) i7-8550U CPU @ 1.80Hz 1.99 GHz processor and 8.00 GB (7.88 GB usable) installed memory (RAM).

4.1 Setup

We present multiple classification language models which cater for the English, Maltese and Maltese-English languages as well as five different social opinion dimensions, namely subjectivity, sentiment polarity, emotion, irony and sarcasm. We train models using state-of-the-art deep neural network models for each of the five opinion dimensions using the Transformer model architecture introduced by Vaswani et al. (Vaswani et al., 2017), which is based on attention mechanisms and is designed to handle sequential data, such as natural language, for NLP tasks like sentiment analysis and text summarisation.

4.2 Handling Imbalanced Data

As reflected in Table 1, the dataset we use is imbalanced. There are several re-sampling techniques (Cateni et al., 2014; More, 2016) for treating the problem of an imbalanced dataset. For our initial

²https://doi.org/10.5281/zenodo.4650232
³https://colab.research.google.com/
⁴https://jupyter.org/
<table>
<thead>
<tr>
<th>Dataset</th>
<th>All</th>
<th>English</th>
<th>Maltese-English and Maltese</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count</strong></td>
<td><strong>Percentage</strong></td>
<td><strong>Count</strong></td>
<td><strong>Percentage</strong></td>
</tr>
<tr>
<td><strong>Subjectivity</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subjective (1)</td>
<td>2591</td>
<td>1713</td>
<td>852</td>
</tr>
<tr>
<td>Objective (0)</td>
<td>3796</td>
<td>3019</td>
<td>744</td>
</tr>
<tr>
<td><strong>Sentiment Polarity</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Negative (0)</td>
<td>1232</td>
<td>775</td>
<td>441</td>
</tr>
<tr>
<td>Neutral (1)</td>
<td>1605</td>
<td>1355</td>
<td>219</td>
</tr>
<tr>
<td>Positive (2)</td>
<td>3550</td>
<td>2602</td>
<td>936</td>
</tr>
<tr>
<td><strong>Emotion</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joy (0)</td>
<td>2636</td>
<td>1976</td>
<td>648</td>
</tr>
<tr>
<td>Trust (1)</td>
<td>363</td>
<td>219</td>
<td>144</td>
</tr>
<tr>
<td>Fear (2)</td>
<td>72</td>
<td>61</td>
<td>11</td>
</tr>
<tr>
<td>Surprise (3)</td>
<td>177</td>
<td>116</td>
<td>60</td>
</tr>
<tr>
<td>Sadness (4)</td>
<td>245</td>
<td>176</td>
<td>67</td>
</tr>
<tr>
<td>Disgust (5)</td>
<td>498</td>
<td>275</td>
<td>216</td>
</tr>
<tr>
<td>Anger (6)</td>
<td>369</td>
<td>238</td>
<td>127</td>
</tr>
<tr>
<td>Anticipation (7)</td>
<td>2027</td>
<td>1671</td>
<td>323</td>
</tr>
<tr>
<td><strong>Sarcasm</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sarcastic (1)</td>
<td>177</td>
<td>101</td>
<td>74</td>
</tr>
<tr>
<td>Not Sarcastic (0)</td>
<td>6210</td>
<td>4631</td>
<td>1522</td>
</tr>
<tr>
<td><strong>Irony</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ironic (1)</td>
<td>329</td>
<td>189</td>
<td>136</td>
</tr>
<tr>
<td>Not Ironic (0)</td>
<td>6058</td>
<td>4543</td>
<td>1460</td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English (0)</td>
<td>4732</td>
<td>4732</td>
<td>100%</td>
</tr>
<tr>
<td>Maltese (1)</td>
<td>299</td>
<td>299</td>
<td>18.73%</td>
</tr>
<tr>
<td>Maltese-English (2)</td>
<td>1297</td>
<td>1297</td>
<td>81.27%</td>
</tr>
<tr>
<td>Other (3)</td>
<td>59</td>
<td>59</td>
<td>0.92%</td>
</tr>
</tbody>
</table>

Table 1: Class distribution for each annotation per dataset

The experiments, we do not address the imbalance or explore whether it influences our classification tasks and if so, which ones. The dataset was divided in a training set of 70%, validation set of 20% and a test set of 10%. The scikit-learn\(^5\) train_test_split function is used to split the sets in a random state.

### 4.3 Models

The following state-of-the-art deep neural network models have been fine-tuned for *subjectivity* (binary), *sentiment polarity* (multi-class), *emotion* (multi-class), *sarcasm* (binary) and *irony* (binary) classification:

- **BERT** (Devlin et al., 2018): A pre-trained model on BookCorpus and English Wikipedia. The BERT-Base uncased, 12-layer, 768-hidden, 12-heads, 110M parameters model is used.
- **DistilBert** (Sanh et al., 2019): A distilled version of the BERT model which is smaller and faster than BERT and is pre-trained on the data. The uncased model which has 40% less parameters than BERT-Base uncased is used.
- **BERTweet** (Nguyen et al., 2020): A large-scale language model pre-trained for English tweets based on the RoBERTa (Liu et al., 2019) pre-training procedure using the same model configuration as BERT-Base. Both bertweet-base (base) and bertweet-covid19-base-uncased (covid-19) models with 135M parameters each are used. The former model is trained on 845M English cased tweets, whereas the latter model is trained on 23M COVID-19 English uncased tweets.

The experiments are carried out using the Hugging Face (Wolf et al., 2019) state-of-the-art Transformer library for Pytorch and TensorFlow 2.0\(^6\). This tool provides general-purpose architectures, such as BERT, RoBERTa and DistilBert for NLP tasks, such as sentiment analysis, where over 32+ pre-trained models are available in 100+ languages. The following hyperparameters are used:

- Optimiser and learning rate scheduler: batch size - 32, Adam (Kingma and Ba, 2014) learning rate - 2e-5, number of epochs - 4, epsilon parameter - 1e-8;
- Method of choosing values and criterion used: Manual tuning based on training and validation loss, learning rates of 5e-5, 3e-5, 2e-5

\(^5\)https://scikit-learn.org/

\(^6\)https://huggingface.co/transformers/
and maximum sentence length of 96, 128 and 256 tokens;

- Fine-tuning classification layer: Rectified Linear Unit (ReLU).

5 Results and Discussion

Results per classification model for each social opinion dimension are presented in Table 2 and further discussed below. Three evaluation metrics are used to measure the classification performance of the fine-tuned models:

- **F1 score weighted**: F1 score is the weighted average of precision and recall. The weighted score calculates the F1 score for each label with their average being weighted by support, that is, the number of true instances for each label. This metric caters for label imbalance.

- **Area Under the Curve Receiver Operating Characteristics (AUC ROC)**: Score shows the model’s true positive rate against the false positive rate and can help you identify how well (score of 1) a model can distinguish between classes.\(^7\)

- **Matthews correlation coefficient (MCC)**: Measures quality of binary and multi-class classifications by taking into account true and false positives and negatives and provides a balanced measure for imbalanced classes.

The following is an overview of the results and some observations.

- **Subjectivity**: For the BERT and DistilBERT models, the training and validation loss converged in epoch 2, whereas both BERTweet models converged in epoch 3. The BERTweet covid19-base-uncased fine-tuned model produced the best performance overall.

- **Sentiment Polarity**: The fine-tuned BERT and DistilBERT models converged in epoch 3, whereas both BERTweet models converged in epoch 4. The BERTweet covid19-base-uncased fine-tuned model also produced the best performance overall.

- **Emotion**: The fine-tuned BERT and DistilBERT models converged in epoch 4, whereas both BERTweet models did not converge by epoch 4 albeit close. An additional experiment showed convergence in epoch 6. In terms of performance, both BERT and DistilBERT fared best overall.

- **Sarcasm**: All fine-tuned models performed similarly in terms of F1 score, with DistilBERT performing best overall. The BERTweet covid19-base-uncased model did not converge in epoch 4 albeit close.

- **Irony**: DistilBERT produced the best results overall, which model converged in epoch 3.

- **Language**: It is interesting to see English-based fine-tuned models adapt to non-English text. This Maltese-English and Maltese subset amounts to only a quarter of the dataset (1596 online posts). Initial results obtained are promising for building language models that are capable of handling code-switched data, which is common practise in countries like Malta. More in-depth experiments and qualitative analysis shall be beneficial to measure the adaptability of the English-based fine-tuned models to code-switched languages, such as Maltese-English.

- **Domain**: A socio-economic dataset (domain specific) has been used, with only 16.75% of the data being off-topic. The results ob-

<table>
<thead>
<tr>
<th></th>
<th>F1 Score</th>
<th>AUC ROC</th>
<th>MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subjectivity</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BERT</td>
<td>0.93</td>
<td>0.983</td>
<td>0.864</td>
</tr>
<tr>
<td>DistilBERT</td>
<td>0.93</td>
<td>0.980</td>
<td>0.851</td>
</tr>
<tr>
<td>BERTweet (base)</td>
<td>0.93</td>
<td>0.970</td>
<td>0.857</td>
</tr>
<tr>
<td>BERTweet (covid19)</td>
<td>0.94</td>
<td>0.975</td>
<td>0.887</td>
</tr>
<tr>
<td><strong>Sentiment Polarity</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BERT</td>
<td>0.85</td>
<td>1 - 0.945</td>
<td>0.748</td>
</tr>
<tr>
<td>DistilBERT</td>
<td>0.83</td>
<td>0.710</td>
<td></td>
</tr>
<tr>
<td>BERTweet (base)</td>
<td>0.86</td>
<td>0 - 0.961</td>
<td>0.772</td>
</tr>
<tr>
<td>BERTweet (covid19)</td>
<td>0.87</td>
<td>2 - 0.964</td>
<td>0.781</td>
</tr>
<tr>
<td><strong>Emotion</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BERT</td>
<td>0.60</td>
<td>3 - 0.935</td>
<td>0.495</td>
</tr>
<tr>
<td>DistilBERT</td>
<td>0.60</td>
<td>0 - 0.913</td>
<td>0.484</td>
</tr>
<tr>
<td>BERTweet (base)</td>
<td>0.58</td>
<td>2 - 0.862</td>
<td>0.478</td>
</tr>
<tr>
<td>BERTweet (covid19)</td>
<td>0.59</td>
<td></td>
<td>0.501</td>
</tr>
<tr>
<td><strong>Sarcasm</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BERT</td>
<td>0.96</td>
<td>0.858</td>
<td>0.073</td>
</tr>
<tr>
<td>DistilBERT</td>
<td>0.96</td>
<td>0.879</td>
<td>0.265</td>
</tr>
<tr>
<td>BERTweet (base)</td>
<td>0.96</td>
<td>0.873</td>
<td>0</td>
</tr>
<tr>
<td>BERTweet (covid19)</td>
<td>0.96</td>
<td>0.792</td>
<td>0</td>
</tr>
<tr>
<td><strong>Irony</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BERT</td>
<td>0.93</td>
<td>0.883</td>
<td>0.179</td>
</tr>
<tr>
<td>DistilBERT</td>
<td>0.93</td>
<td>0.896</td>
<td>0.240</td>
</tr>
<tr>
<td>BERTweet (base)</td>
<td>0.92</td>
<td>0.862</td>
<td>0</td>
</tr>
<tr>
<td>BERTweet (covid19)</td>
<td>0.92</td>
<td>0.887</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2: Results of all the pre-trained models

\(^7\)For the sentiment polarity and emotion multi-class models we only display the maximum AUC ROC score for each respective class.
tain in our preliminary work demonstrate that fine-tuning models to new domains is possible when using deep neural network models.

- The DistilBERT model took less time for training and validation for all five classifiers.
- Even though the dataset is imbalanced, the subjectivity and sentiment polarity models produced good results. However, certain resampling techniques shall help increase the performance of the sarcasm and irony fine-tuned models, which class distribution is very unbalanced as reflected by the MCC. The same also applies to the emotion model for certain classes, such as fear, surprise, sadness.
- Several researchers recommend only 2-4 epochs of training for fine-tuning BERT on a particular NLP task. However, certain multi-class classification tasks with a large number of classes such as the emotion 8-class classification fine-tuned model, might require more than 4 epochs when certain models such as BERTweet are fine-tuned.
- Given that the dataset used contains a mix of newswire comments and tweets, the maximum sentence length in the dataset used is 867. Therefore, more experiments should be carried out using a higher maximum sentence length than the 128 tokens used. However, the high computation power needed for training such deep learning models should be taken in consideration to reduce the carbon footprint in terms of finance and the environmental (Strubell et al., 2019).

6 Conclusions and Future Work

We have leveraged a novel multidimensional and multilingual social opinion dataset in the socio-economic domain to fine-tune neural language models targeting English-Maltese social data for different opinion dimensions, namely subjectivity, sentiment polarity, emotion, sarcasm and irony. Even though our results are a work-in-progress, we have been encouraged by Xia et al. (Xia et al., 2020) to provide multilingual benchmarks which can be further used, evaluated and adapted for low-resourced languages. Research applications for the developed classification models include opinion summarisation and fine-grained opinionated search of each dimension.

Acknowledgments

This work is funded by the ADAPT Centre for Digital Content Technology which is funded under the SFI Research Centres Programme (Grant number 13/RC/2106).

References


Towards an Etymological Map of Romanian

Alina Maria Cristea,1 Anca Dinu,1 Liviu P. Dinu,1
Simona Georgescu,1 Ana Sabina Uban,1,2 Laurentiu Zoicas1

1University of Bucharest
2Universitat Politècnica de València
alina.cristea@fmi.unibuc.ro, ancaddinu@gmail.com
liviu.p.dinu@gmail.com, simona.georgescu@lls.unibuc.ro
ana.uban@gmail.com, laurentiu.zoicas@lls.unibuc.ro

Abstract

In this paper we investigate the etymology of Romanian words. We start from the Romanian lexicon and automatically extract information from multiple etymological dictionaries. We evaluate the results and perform extensive quantitative and qualitative analyses with the goal of building an etymological map of the language.

1 Introduction

Located at crossroads between East and West, the Romanian language presents a kaleidoscopic etymological picture. Originated from Latin, it suffered the influence of many cultures with which the other Romance languages did not have (much or any) contact, hence its physiognomy became, from a certain point on, different from that of its cognate languages (cf. Niculescu (1965, 1978, 1999, 2003)). The Romanian lexicographers, having to deal with this miscellaneous etymological structure of the language, must perform a fairly complicated task which not rarely ends up by giving in to the difficulty of identifying a word’s origin.

Our analysis, based on a computational systematization of the origins of words, aims to evaluate quantitatively and qualitatively Romanian’s etymological composition. We propose a socio-cultural interpretation of the semantic domains most permeable to borrowings from the various languages with which Romanian had a stronger contact, considering that a systematic perspective on the lexicon’s etymological structure, doubled by statistics on the permeability and needs of the various onomasiological fields, may provide clues for future research concerning still unknown etymologies.

1.1 Preliminaries. Peculiarity of Romanian vs. the Other Romance Languages

Apart from its genetically belonging to the Romance linguistic family, the Romanian language shares certain phonological, morpho-syntactic and lexical features with the Balkan languages, as a consequence of its geographical position. For this reason, it was also included in the so-called “Balkansprachbund” or “Balkan language area” (cf. Rosetti (1968)), together with Eastern South Slavic languages (Bulgarian, Macedonian and Serbian), Albanian and Greek.

There are two significant differences between Romanian and the other Romance languages:

1. According to Sala et al. (1988), the fundamental lexical core of Romanian comprises less words inherited from Latin than the other Romance languages (Ro. 30% vs. It. 44%, Fr. 36%, Sp. 40%, Pt. 45%).

2. At the same time, while the Italo-Occidental Romance languages make use, in their basic lexicon, of at least 25% loanwords from Latin (It. 28%, Fr. 27%, Sp. 27%, Pt. 25%), the Romanian language only counts little more than 1% words borrowed directly from Latin; thus, even if we add the 8% Latin words borrowed via French and Italian, the most Eastern Romance language still does not reach the Occidental proportion of the “cultural superstrate” (cf. Reinheimer Ripeanu (2004)).

By combining these two components (inherited and borrowed words from Latin), considering their proportion in the representative lexicon of the Romance languages, we obtain It. 72%, Fr. 63%, Sp. 67%, Pt. 70%, while in Romanian the Latin element hardly reaches 32% (or 39% if we also consider the Latin words penetrated via French and
A reason for this considerable etymological divergence could be, on the one hand, its late integration and early separation from the Roman Empire: conquered at the beginning of the 2nd century, Dacia was left unconnected with the Empire in the second half of the 3rd century. This could explain the lower proportion of inherited words. On the other hand, the different geographical context had a significant influence on the further development of the Romanian language, because, while the Italo-Occidental Romance languages were passing through a period of re-latinisation, massively borrowing words from Latin, the Oriental Latin descendant had strong contact with the Slavic, Greek and Turkish languages, all of which have left deep marks on the Romanian lexicon.

We must also briefly describe here another particularity of the Romanian lexicon, namely the external multiple etymology, defined as “the provenance of a single word from two or more lending languages, at the same time and on the same territory, or in different times and in different territories” (Celac, 2020). This situation resides in multiple internal and external factors that influenced the Romanian language simultaneously, especially during its modernization period (the 19th century). The “cultural loanwords” (i.e., words related to technology, science, cultural life, mostly corresponding to the international vocabulary items, cf. Moroianu (2015); Celac (2020)) could penetrate more or less at the same time from different source-languages, depending on the foreign language that was used as a source in the borrowing process. As the languages that were used as source for the neological enrichment of Romanian are multiple – besides French and Italian we also count Latin, Modern Greek, Russian and German –, it is not infrequent the case where a word has three, four or even five etymologies.

Moreover, one should take into account the dialectal fragmentation of Romanian before its cultural unification and standardization (starting not before the second half on the 19th century), which led to the same situation of multiple-source borrowing, depending on the contact language of each Romanian province: for example, the Romanian speakers in Moldavia would borrow from the Ukrainian language, while Southern Romania would use Bulgarian or Serbian as source-languages. Thus, one and the same Slavic word could have penetrated through different channels, which results as well in multiple etymology.

While the concept of “multiple etymology” is rather unusual for the other Romance languages, this peculiar situation being almost absent in the rest of the Latin descendants, the Romanian language has a significant number of lexical units borrowed more or less simultaneously from various sources, that reach a proportion of almost 18% of the fundamental lexical core (cf. Sala et al. (1988)).

This situation represents one of the main difficulties that Romanian lexicographers have to face. In our approach, we will provide a statistic of words having from one up to six etymologies. It goes without saying that the possibility of errors cannot be overlooked, as many lexicographers have also dealt with this particular Romanian lexical characteristic by placing at the same level several etymologies, whenever they were simply unsure about the immediate origin of a word.

1.2 Romanian Lexicography – A Brief Survey

In this section we offer a brief overview of the main resources one can use for etymological information concerning the Romanian lexicon. We also present the dictionaries we used for this research, explaining the reasons for our choices.

By comparing the lexicographical resources for Romanian with those created for the other main Romance languages (Italian, French, Catalan, Spanish and Portuguese), one can notice the absence of a substantial etymological dictionary of Romanian, equivalent to the lexicographic instruments we can use, for instance, for French (FEW (Wartburg, 1922–2002)), Catalan (DECat (Coromines, 1980–2001)) or Spanish (DCECH (Coromines and Pascual, 1980–1991)).

Despite various attempts to provide reliable etymological dictionaries, the results have been either incomplete (e.g., Etymologicum Magnum Romaniae (Hasdeu, 1886–1898), ceased at the letter B, or Candrea and Densusianu (1907–1914) – comprising only the words of Latin origin, besides not going further than the letter P), or not fully trustworthy (DER, cf. Hristea (2009)). The thesaurus dictionary of Romanian, DA (Pușcariu, 1913–1949) / DLR (Iordan, 1965–2010), is not consistent in the etymological descriptions it offers: while the first volumes, A-De and F-Lojnită (Pușcariu, 1913–1949), offer solid etymological descriptions, the remaining volumes (Iordan, 1965–2010) – reduce
the etymological explanations to a minimum. The ongoing project of a new complete etymological dictionary, DELR ((Academia Română, 2011–), covering so far the letters A–C), has been criticized not only for punctual shortcomings (cf. Celac (2012)), but for its whole design, being destined only to review the tradition of the etymological research on the lexemes (cf. Ernst (2013); Schweickard (2013)).

Somewhat more reliable sources for Romanian etymology, despite not having been designed to meet this purpose, but as explanatory dictionaries of the language, are Ţăineanu (1929), Scriban (1939) and DEX ((Academia Română, 1996 [1975]), second edition bis 2009, second edition ter 2012).

Since one of the requirements for this research was the use of complete and consistent sources that are, at the same time, available online, we resorted to the following dictionaries, listed below in order of their priority: DEX ’16, DEX ’09, DER (Cioreanescu, 1966), Scriban (1939), Ţăineanu (1929), DEX ’12, DEX ’98, DEX ’96, DEX ’84, DEX ’75, DEX-S (Academia Română, Institutul de Lingvistică din Bucuresti, 1988), DN (Marcu and Maneca, 1986), DLRLV (Costinescu et al., 1987). The order of the sources in our analysis was shaped according to their relative reliability, which was established following the empirical observations regarding the accuracy of the data provided.

2 Extracting and Processing the Data

In this section we describe our procedure for automatically extracting and processing etymological information for the Romanian lexicon.

<table>
<thead>
<tr>
<th>Dictionary</th>
<th>Match</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEX ’09</td>
<td>– Din #fr.# @abat-jour.@</td>
</tr>
<tr>
<td>DN</td>
<td>[…] / &lt; #fr.# Sabat-jour$</td>
</tr>
<tr>
<td>Scriban</td>
<td>(fr. Sabat-jour.$ […]</td>
</tr>
<tr>
<td>DER</td>
<td>&lt; #Fr.# Sabat-jour,$</td>
</tr>
</tbody>
</table>

Table 1: Examples of different formats for representing etymological information in dictionaries covered by dexonline for the Romanian word abajur (meaning lampshade), which is borrowed from the French word abat-jour.

2.1 Data

We identify the etymologies and etymons of Romanian words using dexonline,¹ a machine-readable dictionary which aggregates information from over 30 Romanian dictionaries. Some of these are restricted by license and copyright, but others are publicly available. Dexonline provides the public data as an SQL dump, which we import in a local database server for querying.² By parsing the definitions from the etymological dictionaries listed in the previous section, we automatically extract information regarding words’ etymologies. The definitions are partly formatted, with different formats for different dictionaries. We extract the relevant information using regular expressions. In Table 1 we provide examples of different formats for representing etymological information in dexonline.

When more options are possible for explaining a word’s etymology, dexonline provides several hypotheses. We account for all the given alternatives, enabling our method to issue more accurate results, both when a dictionary considers a word to have multiple etymology (e.g., DEX ’09 provides both French vérisme and Italian verismo as etymologies of verism, meaning “a literary and musical movement developed at the end of the 19th century”) and when different dictionaries provide different languages of origin (e.g., DEX ’09 provides Russian koleaska as etymology for calească (meaning carriage), while Scriban provides French calèche as etymology for the same word).

We introduce the order of priority mentioned in Section 1.2 in case different dictionaries provide different etymons (or different orthographic forms of the same etymon) for a certain word and language of origin (e.g., DEX ’09 provides French abattis as etymology for abatică (meaning abatis), while DN spells the word abbatis).

In cases of homonymy, we take into account all the separate dictionary entries. By homonyms we mean words that have the same form, but different origins (e.g., lac1 meaning lake, and lac2 meaning lacquer; according to DEX ’09, the first lexeme is inherited from the Latin word lacus, while the second one is borrowed from the German word Lack; the form coincidence derives from the historical phonetics of Romanian). All the values reported henceforth refer to words as conjunctions between a phonetic form and a conceptual content, taking into account their origin and history, and not only as raw word forms.

¹https://dexonline.ro

²We use the database backup available on January 17, 2021.
2.2 Processing

We employ several post-processing steps for the etymological information, mainly for cleaning and normalization. For etymons, we keep both the processed forms and the original ones, for future reference. We provide below some processing rules along with motivations.

For extracted source languages:

- Grouping together different abbreviations for source languages used by different dictionaries (e.g., \textit{tc}, \textit{tur}, \textit{turc}, \textit{turk} all refer to Turkish).

- Conflating different periods of some languages (e.g., we group \textit{vlat}, \textit{mlat}, \textit{nlat} – Old, Medieval, Neo-Latin under Latin), while keeping separated languages such as Old Slavic vs Slavic or Ancient Greek vs Neo-Greek.

For extracted etymons:

- Removing some diacritical symbols that mark the stressed syllable or vowel length and are not regularly rendered in the source language (e.g., Italian \textit{ab`ate} becomes \textit{abate} after removing the diacritical mark of the stress; Latin \textit{abbatt`ere} becomes \textit{abbattere} after removing the diacritical mark of a short vowel, which shows that the stressed syllable is the antepenultimate).

- Replacing the rough breathing mark ‘ with the letter \textit{h} for Greek etymons. This diacritical mark is rendered, in the transcription from Ancient Greek into Latin, by the letter \textit{h}, and we apply the same transformation (e.g., the Greek etymon \textit{onomophonos} of the word \textit{omofon} (meaning \textit{homophonous}) becomes \textit{homophonos} after removing the stress mark and replacing the rough breathing mark).

- Removing endings for the oblique cases of Latin or Greek etymons (e.g., \textit{marmor, -oris} for \textit{marmur} (meaning \textit{marble}), Neo-Greek \textit{`aroma}, \textit{ar`omados} for \textit{aromat} (meaning \textit{aromatic}) or secondary forms provided (e.g., Latin \textit{adnotare}, \textit{annotare} for a \textit{adnota} (meaning to annotate), French \textit{ballerin}, \textit{ballerine} for \textit{balerin} (meaning \textit{ballet dancer})).

- Removing the asterisk symbol that marks untested etymons (e.g., the Latin etymon *\textit{conquerire} of a \textit{cuceri}, meaning \textit{to conquer}).

- Removing letters provided between round or square brackets. The former represent the spelling from the cultured language for Latin (e.g., \textit{invol(u)tas}, etymon of \textit{invol}, meaning \textit{abundant}), but Romanian inherited words do not originate from the cultured language. The latter have different meanings, such as reconstructing an intermediary form of the word (e.g., Latin \textit{ecceum-\[i\]loc}), but in any case the information is not relevant for this study.

- Filtering out proper names, since they are not relevant for this study.

<table>
<thead>
<tr>
<th>Source language</th>
<th>#words</th>
<th>#verbs</th>
<th>#adjectives</th>
<th>#nouns</th>
</tr>
</thead>
<tbody>
<tr>
<td>French</td>
<td>35,511</td>
<td>2,533</td>
<td>8,219</td>
<td>23,610</td>
</tr>
<tr>
<td>Latin</td>
<td>9,313</td>
<td>1,203</td>
<td>2,215</td>
<td>5,302</td>
</tr>
<tr>
<td>Italian</td>
<td>3,358</td>
<td>384</td>
<td>471</td>
<td>1,960</td>
</tr>
<tr>
<td>German</td>
<td>2,767</td>
<td>73</td>
<td>300</td>
<td>2,331</td>
</tr>
<tr>
<td>English</td>
<td>2,064</td>
<td>41</td>
<td>253</td>
<td>1,700</td>
</tr>
<tr>
<td>Greek</td>
<td>1,754</td>
<td>1</td>
<td>380</td>
<td>1,141</td>
</tr>
<tr>
<td>Turkish</td>
<td>1,293</td>
<td>3</td>
<td>73</td>
<td>1,092</td>
</tr>
<tr>
<td>Slavic</td>
<td>1,155</td>
<td>236</td>
<td>86</td>
<td>803</td>
</tr>
<tr>
<td>Neo-Greek</td>
<td>1,026</td>
<td>54</td>
<td>51</td>
<td>836</td>
</tr>
<tr>
<td>Russian</td>
<td>896</td>
<td>9</td>
<td>62</td>
<td>777</td>
</tr>
<tr>
<td>Old Slavic</td>
<td>836</td>
<td>1</td>
<td>95</td>
<td>652</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>650</td>
<td>60</td>
<td>33</td>
<td>533</td>
</tr>
<tr>
<td>Hungarian</td>
<td>622</td>
<td>50</td>
<td>35</td>
<td>472</td>
</tr>
<tr>
<td>Serbian</td>
<td>532</td>
<td>48</td>
<td>20</td>
<td>428</td>
</tr>
<tr>
<td>Ukrainian</td>
<td>270</td>
<td>19</td>
<td>10</td>
<td>235</td>
</tr>
<tr>
<td>Spanish</td>
<td>220</td>
<td>1</td>
<td>10</td>
<td>193</td>
</tr>
<tr>
<td>Polish</td>
<td>181</td>
<td>1</td>
<td>7</td>
<td>161</td>
</tr>
<tr>
<td>Ruthenian</td>
<td>151</td>
<td>3</td>
<td>6</td>
<td>124</td>
</tr>
</tbody>
</table>

Table 3: The number of Romanian words that originate from each source language. We report only the languages from which at least 100 words originated.

3 Quantitative Analysis

In Table 2 we report the number of Romanian words having zero, one or multiple etymologies identified automatically. 48,887 words out of a total of 94,244 words have at least one automatically
identified etymology and this set will constitute our data from now on. In Table 3 we report the number of Romanian words that originate from each source language and how many of these words are verbs, adjectives and nouns. In Figure 1 we illustrate the distribution of Romanian words of different etymology, by proportion mapping. The bigger the bullet, the more Romanian words originated from the language in that geographical region. Note that we do not dispose of dated etymologies and so we lack the diachronic dimension; thus, different languages or different evolution stages of the same language are represented on the same territory (e.g., the two circles from Italy represent Latin and Italian and the three circles located in present day Bulgaria correspond to Old Slavic, Slavic and Bulgarian).

We also compared the data we obtained with the information we knew concerning the fundamental lexical core from Sala et al. (1988), namely 7.5% French borrowings and 30% Latin inherited words. We easily notice not only an inverted ratio between the quantity of words of Latin origin and those originated from French, but also a hugely expanded proportion of French borrowings. About 38% of the whole lexicon and almost 73% of the words that have at least one etymology attested in the dictionaries have French origin (versus only 7.5% of the representative lexicon), while the quantity of words of Latin origin (most of them inherited) hardly reaches 9% of the whole lexicon and 19% of the words that have at least one etymology attested in the dictionaries (versus the proportion of 30% for the fundamental lexical core). The gap is explainable by the distinction between the basic, common lexical core (covering 80% of everyday speech) and the cultured lexicon and specialised terminology, developed in the last century by massively borrowing lexical items from French.

Sorting the borrowings of each language by parts of speech highlights the significant quantitative breach between the nominal parts (by far the majority) and the verbal ones. But, while the inherited lexicon shows a ratio of 1 verb to 6 nominal parts (noun+adjective), the borrowing process has considerably enriched the quantity of nominal parts of speech, in detriment of the verbal ones: e.g., the French borrowings encapsulate a ratio of 1 verb to 12 nominal parts, the English ones display a ratio of 1 verb to 48 nominal parts, and the Turkish loanwords enclose a correlation of 1 verb to 388 nominal parts. This situation allows a deeper insight into the language structure, showing that expressing an action, state or occurrence requires a higher degree of internalized lexicon or of acquaintance with the language: we can deduce, on the one hand, that the speakers are able to express their experiences by using a fairly small number of verbs, but need to constantly increase the amount of nouns to designate the new objects they observe or concepts they acquire; on the other hand, morphosyntactic restriction may also play a part: while a nominal part of speech is easily adaptable to the morphological system of Romanian, the complex verbal conjugation may impede its immediate adoption. Also, it seems that the more related the source language is to Romanian, the easier is the morphosyntactic adaptation of verbs, which might explain the above ratio order, French, English, Turkish.

By classifying the lexicon in parts of speech, we also notice a shortcoming in the Romanian lexicography, namely the inconsistency of lexicographers when establishing the period when a word of Slavic origin entered the Romanian language: from this categorization it results that only one verb was borrowed from Old Slavic, while more than 200 come from Slavic. It is, however, evident that many fundamental verbs of Slavic origin have penetrated during the period of early contact between the two communities, thus, originate from Old Slavic (e.g., a iubi (meaning to love), a citi (meaning to read), a gresi (meaning to make a mistake)). In this case, we only highlight a terminological misunderstanding.

In order to quantify the resemblance between Romanian words and their etymons, for different source languages, we compute the edit distance (Levenshtein, 1965) for \{word, etymon\} pairs, using the post-processed etymon form (see Section 2.2). The edit distance counts the minimum number of operations (insertion, deletion and substitution) required to transform one string into another. We use a normalized version of this metric, dividing the edit distance by the length of the
longest string. The obtained values are between 0 and 1; the lower the values, the closer the Romanian words are to their etymons. In Figure 2 we report the average edit distance between the Romanian words and their etymons, per language. Overall, Romanian words borrowed from English are closest to their etymons. For 990 out of 2,064 words with English etymology, the edit distance is 0, meaning that those Romanian words have not undergone any transformations when entering the language (e.g., marketing, management, avocado). For Latin, 633 out of 9,313 words are identical to their etymon (e.g., vultur, meaning vulture).

Figure 2: Average normalized edit distance between Romanian words and their etymons for the top 10 languages from Table 3.

4 Qualitative Analysis

4.1 Analysis of Lexicographical Errors

In order to evaluate our automatic method for extracting etymologies, we excerpt a sample of 1,000 words. We manually determine the etymologies of the words in the sample using the web interface of dexionline, we compare these results with the automatically obtained etymologies, and we report an accuracy of 99.2%.

The main error source is the recording of erroneous etymologies in the dictionaries. One of the most common errors is to consider the ultimate origin (either Latin or Ancient Greek) as the immediate etymology of a Romanian word, without taking into account its form and sometimes meaning, which point to a different source language. To take just an example, apotr´opaion meaning “magic remedy to ward off evil” is considered to be directly originated from Ancient Greek apôtro´paoion, but neither the form nor the meaning allows such supposition: on the one hand, it is not usual for a word borrowed as a proparoxytone (a word stressed on the ante-penultimate syllable) to become an oxytone (stressed on the ultimate syllable) in Romanian, on the other hand, the Greek word functioned as an adjective, apotrópaios, whose meaning tute-
lary / expiatory / abominable does not precisely match the Romanian significance. Nonetheless, if we take a look at the European modern languages, we can easily find the German lexeme Apotrop¨aum, meaning exactly “magic remedy to ward off evil”, as a term circumscribed to archaeology, both formally and semantically able to account for the Romanian word. Thus, it would be correct to indicate the German noun as the immediate origin of the Romanian word, and not the Ancient Greek adjective, with which it only has a distant connection.

A quite frequent error consists of almost automatically labelling a “cultural loanword” as French. For instance, the origin of Ro. helipot (meaning helipot) is attributed to a nonexistent French word “hélipot”. Similarly, certain dictionaries invent a French word acquisiteur (for acq´ereur) in order to explain Ro. achizitor (meaning acquirer). A similar example is that of Ro. national (meaning national), explained as a borrowing from Latin nationalis (to which the French word national is added, by virtue of the concept of multiple etymology). Nonetheless, the supposed Latin word nationalis is not documented in Latin, the concept being a modern one.

4.2 A Semantic Insight into the Romanian Lexicon’s Structure

In this section we provide an analysis of the etymo-
logical composition of the Romanian lexicon based on semantic fields.

We start by building a list of conceptual domains, based on the Romanian linguistic atlases (Puscaru, 1938–1942; Petrovici, 1956–1972), which provide a list of semantic fields that covers the vocabulary, containing as well the most usual terms belonging to each of these onomasiological fields. We select a subgroup of these, and merge a few together, resulting in a final list of 10 semantic fields. We then manually extract a selection of prototypical terms for each of the resulted groups, on average 36 terms per group. We employ these terms as seeds for automatically populating the semantic clusters, using semantic similarity metrics based on word embeddings, a standard method for measuring lexical semantic similarity in the field of computational
analysis of semantic change. In our study, we make use of word embeddings computed using the FastText algorithm, pre-trained on Wikipedia for the top six languages Romanian borrowed from. The vectors have 300 dimensions and were obtained using the skip-gram model described by Bojanowski et al. (2016) with default parameters. These embeddings have previously been used in studies on semantic similarity of cognate sets in Romance languages (Uban et al., 2019, 2021). To group the terms in our dataset into the different semantic fields, we apply a KNN classifier (k=7) trained on the pre-defined list of semantic groups and prototypical terms. We then retrieve for each semantic cluster the distribution of etymologies for the words it contains. In Table 4 we show the top languages found in the etymologies of words belonging to each cluster.

One can easily observe in Table 4 that in 9 out of 10 semantic domains the first 3 source languages are invariably French, Latin, and Italian, precisely in this order. In 6 out of 10 onomasiological fields, the fourth position is occupied by a Germanic language (either German or English), while in 2 cases it is the Greek language holding this position. In 8 out of 10 domains, at least one Slavic language is represented among the first 8 source languages. It is also noteworthy that the Slavic (probably Old Slavic, see the comment above in Section 3) is the third most represented language in the onomasiological field of animals, and the fourth in the domain of agriculture and fifth in personality / emotions. The Turkish language reaches its highest position (the sixth) in the semantic field of food and drink, which reflects the predominance of trade relations between the two communities. The constant presence of French and Italian (putting aside Latin, which is mostly the source for inherited, not borrowed words) as top source languages in the borrowing process, clearly shows that the genetic relations, on the one hand, and the cultural connections, on the other hand, prevail over the geographical contiguity in the selection of source languages for the lexical enrichment.

5 Conclusions

For historical, geographical and linguistic reasons, Romanian presents a complex lexicographic picture, especially in terms of etymology. While reliable etymological dictionaries for Romanian are still missing, we proposed a computer-assisted etymological analysis doubled by a linguistic manual verification and interpretation, using the available dictionaries via dexonline. The comparison between the obtained data and previous knowledge about the Romanian fundamental lexicon revealed an inverted proportion between French and Latin and a surprisingly high percentage of French borrowings. We visualized the Romanian etymologies per source language on a geographic map and we also minded their part of speech proportions and interpreted them. Error analysis showed that the automatic extraction was performed with high accuracy, while the remaining errors are due to erroneous etymologies from the dictionaries. Finally, we experimented with the etymological composition of the Romanian lexicon based on semantic fields. Starting from a list of conceptual domains, adapted from Romanian linguistic atlases, we automatically obtained 10 onomasiological fields containing Romanian words in our dataset and their etymologies. For each of these categories, we ordered the source languages and interpreted the results from a socio-cultural and historical point of view.

Ethics Statement

All our data are extracted from publicly available sources. There are no ethical issues in our work.
Acknowledgments
We would like to thank the reviewers for their helpful comments. All authors contributed equally to this work. This research is supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI UEFISCDI, project number 108, COTOHILI, within PNCDI III.

References


August Scriban. 1939. Dictionarul limbii românesti. Institutu de Arte Grafice “Presă Bună”.


A Syntax-Aware Edit-based System for Text Simplification

Oscar M. Cumbicus-Pineda,
Ixa group and Carrera de Computation
UPV/EHU and UNL
ocumbicus001@ikasle.ehu.es

Itziar Gonzalez-Dios, Aitor Soroa
Ixa group, HiTZ center
University of the Basque Country (UPV/EHU)
{itziar.gonzalezd,a.soroa}@ehu.eus

Abstract
Edit-based text simplification systems have attained much attention in recent years due to their ability to produce simplification solutions that are interpretable, as well as requiring less training examples compared to traditional seq2seq systems. Edit-based systems learn edit operations at a word level, but it is well known that many of the operations performed when simplifying text are of a syntactic nature. In this paper we propose to add syntactic information into a well known edit-based system. We extend the system with a graph convolutional network module that mimics the dependency structure of the sentence, thus giving the model an explicit representation of syntax. We perform a series of experiments in English, Spanish and Italian, and report improvements of the state of the art in four out of five datasets. Further analysis shows that syntactic information is always beneficial, and suggest that syntax is more helpful in complex sentences.

1 Introduction
Automatic Text Simplification (ATS) aims to reduce the linguistic complexity of a text for a certain target audience. ATS is useful for people learning foreign languages, suffering cognitive disabilities, or with low literacy level. The area of application of ATS is however not restricted to humans, as it has also been used to improve automatic NLP tasks such as parsing (Chandrasekar et al., 1996a), summarization (Beigman Klebanov et al., 2004; Silveira and Branco, 2012), semantic role labeling (Vickrey and Koller, 2008; Woodsend and Lapata, 2017), information extraction (Evans and Orasan, 2019) and machine translation (Gerber and Hovy, 1998; Štajner and Popovic, 2016; Hasler et al., 2017), among others.

ATS is often cast as a machine translation task, where the system receives complex sentences as input, and produces sentences that are simpler yet maintaining the original meaning. While early ATS systems were rule-based, statistical or hybrid (Saggion, 2017), in the last years neural network based ATS approaches have also been proposed (Alva-Manchego et al., 2020). In particular, sequence-to-sequence (seq2seq) neural models (Sutskever et al., 2014; Nisioi et al., 2017; Zhang and Lapata, 2017a) have shown to obtain state-of-the-art results. Such systems are trained on parallel corpora comprising pairs of complex/simple sentences, and implicitly learn the simplification rewrites needed to convert complex sentences into simpler ones.

Neural seq2seq systems are usually black boxes that are trained on an end-to-end fashion. As a consequence, built models are usually very difficult to interpret, and offer little control or hints that explain why a particular input word sequence has been rephrased. Edit-based ATS systems try to overcome this limitation by learning the transformations required to convert complex sentences into their simpler counterparts. The set of transformations is limited and known beforehand, and usually comprise edit operations such as delete, removal or lexical substitution (Alva-Manchego et al., 2017; Dong et al., 2019; Kumar et al., 2020). Because the set of allowed operations is restricted, the search space is considerably reduced. As a consequence, edit-based models are usually sample efficient and require less training examples compared to traditional seq2seq systems (Mallinson et al., 2020; Omelianchuk et al., 2021).

Edit-based ATS systems learn edit transformations at a word level, but often those operations are applied to whole phrases. Besides, systems need to capture long range relations among words, such as syntactic and phrase structures. For example, in the English sentence presented in Table 1 there is a long subject “Dry air wrapping around the southern periphery of the cyclone” that causes the...
Dry air wrapping around the southern periphery of the cyclone eroded most of the deep convection by early on September 12. In this sense, the president of the National Hunting Office, Juan Antonio Sarasketa, assured that this regulation project makes it impossible to possess and use weapons, (…) the president of the National Hunting Office, Juan Antonio Sarasketa, assured that this regulation project makes it impossible to possess and use weapons, (…) the president of the National Hunting Office, Juan Antonio Sarasketa, assured that this regulation project makes it impossible to possess and use weapons, (…) the president of the National Hunting Office, Juan Antonio Sarasketa, assured that this regulation project makes it impossible to possess and use weapons, (…)

Table 1: Sentences with long sentences (English) or subordination clauses (Spanish) cause ATS to fail.

<table>
<thead>
<tr>
<th>Version</th>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orig. (EN)</td>
<td>Dry air wrapping around the southern periphery of the cyclone eroded most of the deep convection by early on September 12.</td>
</tr>
<tr>
<td>Aut.</td>
<td>Dry air wrapping around the southern periphery of the cyclone.</td>
</tr>
<tr>
<td>Orig. (ES)</td>
<td>En este sentido, el presidente de la Oficina Nacional de la Caza, Juan Antonio Sarasketa, aseguró que este proyecto de reglamento imposibilita la posesión y uso de armas. (…) el presidente de la oficina nacional de la caza, juan antonio sarasketa, aseguró que el gobierno de cine de emigrantes</td>
</tr>
<tr>
<td>Aut.</td>
<td>The president of the National Hunting Office, Juan Antonio Sarasketa, assured that this regulation project makes it impossible to possess and use weapons, (…) The president of the National Hunting Office, Juan Antonio Sarasketa, assured that this regulation project makes it impossible to possess and use weapons, (…) The president of the National Hunting Office, Juan Antonio Sarasketa, assured that this regulation project makes it impossible to possess and use weapons, (…) The president of the National Hunting Office, Juan Antonio Sarasketa, assured that this regulation project makes it impossible to possess and use weapons, (…)</td>
</tr>
</tbody>
</table>

The contributions of this paper are the following:

- We present a method to integrate syntactic information into an edit-based seq2seq system.
- The results confirm our intuition that syntactic information is useful for ATS systems. The new system surpasses previous state of the art in four out of five datasets. We conduct an ablation study that shows the effect of removing the syntactic information from the system.

This paper is structured as follows: in Section 2 we present the work done with dependencies, graphs and recent ATS systems; in Section 3 we describe our approach, in Section 4 our experiments and in Section 5 the results, we conclude and outline the future work in Section 7.

2 Related Work

Sentence simplification (SS) aims to reduce a sentence’s reading complexity while preserving its meaning. Recently, systems based on neural networks are gaining attention in ATS. For more information about previous works in ATS the interested reader is referred to these works (Shardlow, 2014; Siddharthan, 2014; Saggion, 2017; Alva-Manchego et al., 2020; Sikka et al., 2020). In this section, we focus on work that are based on dependencies and on one of the recently developed techniques, which is based on editing to simplify the text.

Dependency based simplification has proven to be effective in rule based and statistical simplification to analyze the texts and induct rules in the rule-based and hybrid architectures of ATS systems (indeed, the first module was a parser and many works have relied on dependency parsers) (Chandrasekar et al., 1996b; Woodsend and Lapata, 2011; Siddharthan, 2011; Klerke and Søgaard, 2013; Aranzabe et al., 2013; Saggion et al., 2015) as well as to perform tree transformations together with statistical machine translation (Zhu et al., 2010).

However, in the last years neural approaches have gained interest in ATS and SS research. The most popular framework has been the seq2seq models, which mainly rely on RNN and transformer based architectures (Nisioi et al., 2017; Zhang and Lapata, 2017b). While the majority of ATS systems are supervised, some works have obtained good results using unsupervised approaches (Surya et al., 2019; Martin et al., 2020b).

One of the main drawback of the aforementioned approaches is the lack of interpretability, i.e., the extend to which a system can explain in a meaningful way why certain actions have been performed. Edit-based systems (Alva-Manchego et al., 2017; Dong
et al., 2019; Kumar et al., 2020) offer a solution to the interpretability problem, as they directly learn a sequence of edit operations that convert complex sentences into simpler ones. Alva-Manchego et al. (2017) present a machine translation system that predicts three explicit simplification operations (keep, replace and delete) in aligned pairs of complex-simple sentences. Dong et al. (2019) propose EditNTS, a neural programmer/interpreter that learns to generate edit operations (add, keep, and delete) in a sequential fashion. Kumar et al. (2020) design a scoring function that measures the quality of a candidate sentence based on the fluency, simplicity, and meaning preservation and generate the simplified candidate sentences by iteratively editing the given complex sentence. The operations they take into account are removal, extraction, reordering and substitution.

Syntactic information has been previously used in rule-based systems, for instance, as an indicator to identify the complexity of sentences (Evans and Orasan, 2018). On deep learning systems, graph convolutional networks (GCN) over dependency trees is a usual method that leverages syntactic information into the models, and captures long-range syntactic relations among words. GCNs generalize the convolution operation usually applied in images to arbitrary graphs, and allow to refine information associated to nodes according to the information in the neighbor nodes (Kipf and Welling, 2017). They have been successfully used in NLP tasks such as semantic role labelling (Marcheggiani and Titov, 2017), information extraction (Zhang et al., 2018) and aspect-based sentiment analysis (Wang et al., 2020). Contemporaneous to this work, Zhe Lin (2021) use semantic information in seq2seq systems by including in the graph of the source sentence the predicate-argument relations between content words in a sentence.

3 A Syntax-Aware ATS System

Our system is based on EditNTS, an edit-based system that has obtained state of the art results on many datasets. We start by briefly describing EditNTS. Then, we describe the graph convolutional network that leverages syntactic information derived from dependency trees. Finally, we describe how to integrate the syntactic module into EditNTS.

3.1 EditNTS

We start by briefly describing the EditNTS system, and refer the reader to Dong et al. (2019) for a more detailed description. Let $x = x_1, \ldots, x_{|x|}$ be a complex input sentence and $y = y_1, \ldots, y_{|y|}$ its simplified version. EditNTS learns to produce a series of edit operations $z = z_1, \ldots, z_N$ which, applied over the input sentence $x$, produces $y$. Each edit operation $z_i$ is one of $\{\text{ADD}(w), \text{KEEP}, \text{DELETE}\}$. EditNTS contains an encoder, decoder and interpreter modules, which are described as follows:

**Encoder**

The encoder transforms the input sequence $x$ into a sequence of output and hidden representations ($o_i$ and $h_i$):

$$o_i, h_i = \text{LSTM}(x_{1:i-1})$$  \hspace{1cm} (1)

where $x_i$ is the concatenation of the embedding of the word $x_i$ and the embedding corresponding to the POS tag of $x_i$.

**Decoder**

The decoder receives the input from the encoder, and predicts the next edit label $z_t$ for each timestep $t$. Internally, it contains two recurrent networks that represent the edit operations and the output words produced so far:

$$o_{t}^{\text{edit}} = \text{LSTM}(z_{1:t-1} \mid h_{|x|})$$

$$o_{t}^{y} = \text{LSTM}(y_{1:t-1} \mid h_{|x|})$$

where $z$ are the embeddings of the edit operations and $y$ are the output embeddings\(^1\). The decoder also uses an attention mechanism between the current edit operation and the input words. Let $O$ and $O^{\text{edit}}$ be the matrices whose rows are the output vectors for the encoder and edit recurrent networks, respectively\(^2\). The attention mechanism is defined as follows:

$$E = \text{softmax}(KO^{\text{edit}}O^T)$$

$$c_t = E_tO$$

where $K$ is a parameter learned by the model. The decoder predicts the next edit label $z_t$ using a sequence of linear layers and activation functions. The input of the linear layer $i_t$ is a concatenation

\(^1\)The embedding matrix is shared between the encoder and decoder.

\(^2\)That is, $O_t = o_t$ and $O_t^{\text{edit}} = o_t^{\text{edit}}$. Through the paper we use the notation $M_i$ to represent the $i$th row of matrix $M$.  

331
of the output representation of the input word currently edited $o_{ki}$, the output representation of the previously generated edit labels $o_{j}^{\text{edit}}$, the representation of the previous generated words $o_{j}$ and the attention vector $c_{i}$:

$$i_{t} = [o_{ki}; o_{j}^{\text{edit}}; o_{j}; c_{i}]$$

$$z_{t} = \text{softmax}(V^{t}(\tanh(Vh_{t})))$$

again, $V$ and $V'$ are parameters learned by the model.

**Interpreter**

The interpreter applies the predicted edit operation $z_{t}$ on current word $x_{ki}$ and produces a new word $y_{ji}$.

### 3.2 Graph Convolutional Network Module

To leverage syntactic information into EditNTS, we first compute the dependency tree of the complex sentence $x$, which is represented as an undirected graph. The nodes of the graph are the words in $x$, and the edges represent syntactic relations among them. The graph is represented as an adjacency matrix $A$, where $A_{ij} = 1$ if an edge between nodes $i$ and $j$ exist.

We then apply a series of graph convolutional operations over the syntactic graph using a Graph Convolutional Network module. The module contains a series of $L$ linear layers, each one applying a convolution operation over the graph:

$$h^{l+1} = \text{ReLU}(\tilde{D}^{\frac{1}{2}}AD^{\frac{1}{2}}h^{l}W^{(l)})$$

where ReLU is the linear rectifier activation function, $\tilde{A} = A + I_N$ is the adjacency graph with added self connections, $D_{ii} = \sum j A_{ij}$ is the degree matrix and $W^{(l)}$ is a layer specific parameter to be learned. $h^{0}$ is the input of the graph, a matrix that assigns an embedding to each vertex in the graph. At each level, the convolution operation aggregates the embeddings of neighbor nodes to produce new embeddings that implicitly encode the structure of the underlying graph.

### 3.3 Augmenting EditNTS with syntactic information

In our final system the GCN initial input $h^{0}$ is initialized with the encoder outputs $o_{t}$ of each word in the sentence, and the corresponding adjacency graph derived from the dependency tree. The output $h^{L}$ of the GCN module is then combined with the original $o_{t}$ vectors as a residual connection, which is then passed to the EditNTS decoder (see Figure 1).

### 4 Experiments

In this section we describe the experiments performed within this work. We start by describing the datasets used for training and testing the system, followed by the experimental setting, which includes a description of the metrics used to evaluate the models.

#### 4.1 Datasets and parameters

We experiment our approach on ATS datasets from three languages: English, Spanish and Italian. The datasets used for each language are the following:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Lang</th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wikilarge</td>
<td>EN</td>
<td>296,402</td>
<td>2,000</td>
<td>359</td>
</tr>
<tr>
<td>Newsela</td>
<td>EN</td>
<td>94,208</td>
<td>1,129</td>
<td>1,076</td>
</tr>
<tr>
<td>Simplex</td>
<td>ES</td>
<td>574</td>
<td>143</td>
<td>693</td>
</tr>
<tr>
<td>Newsela-es</td>
<td>ES</td>
<td>50,301</td>
<td>2,794</td>
<td>2,795</td>
</tr>
<tr>
<td>Italian</td>
<td>IT</td>
<td>29,260</td>
<td>1,475</td>
<td>1,475</td>
</tr>
</tbody>
</table>

Table 2: Sizes (number of sentences) of the datasets used in the experiments

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Sent. Length</th>
<th>Sent. Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg.</td>
<td>Std</td>
</tr>
<tr>
<td>Wikilarge</td>
<td>26.78</td>
<td>13.13</td>
</tr>
<tr>
<td>Newsela</td>
<td>19.96</td>
<td>11.81</td>
</tr>
<tr>
<td>Simplex</td>
<td>26.79</td>
<td>10.81</td>
</tr>
<tr>
<td>Newsela-es</td>
<td>16.79</td>
<td>6.01</td>
</tr>
<tr>
<td>Italian</td>
<td>20.23</td>
<td>9.01</td>
</tr>
</tbody>
</table>

Table 3: Corpora statistics (train). Sentence length measures the number of tokens and the depth shows the depth of the dependency tree.
English We used two datasets for English, WikiLarge/TurkCorpus and Newsela. WikiLarge is one of the most used to train ATS systems and was built by automatically aligning sentences belonging to the same article in English Wikipedia and Simple English Wikipedia. We used the split provided by Zhang and Lapata (2017a) for training and development, with a total of 296,402 and 2,000 sentences, respectively. Following usual practice, we tested the models trained on WikiLarge using the TurkCorpus dataset (Xu et al., 2016), which contains eight manually generated reference simplifications for 359 sentences. Newsela consists of 1130 news articles that were rewritten four times at different complexity levels. We used the train/development/test splits from (Xu et al., 2015), containing 94,208/1129/1076 sentences respectively.

Spanish We used two datasets for Spanish, Simplext and Newsela-es. The Simplext corpus contains 200 news texts from different domains, that were manually simplified (Saggion et al., 2015). We use the splits provided by Martin et al. (2020b) with 574/143/693 sentences for training, development and test. Newsela-es is similar to its English counterpart, we used the splits from Palmero Aprosio et al. (2019) and comprises 50,301/2,794/2,795 sentences for train/dev/test.

Italian For Italian we use the documents provided by Palmero Aprosio et al. (2019), a corpus containing 32,210 complex-to-simple pairs sentences that were obtained by merging three available data sets: the SIMPITIKI corpus (Tonelli et al., 2016), the corpora Terence and Teacher (Brunato et al., 2015), and a subset of the PaCCESS-it corpus (Brunato et al., 2016).

Table 2 shows the size of each dataset, and in Table 3 we present the average and standard deviation of the sentence length and depth for each corpus, for both the complex and simple sentences. The sentence length measures the number of tokens of each sentence and the depth shows the maximum depth of the dependency trees.

4.2 Experimental settings

We tokenized and syntactically analyzed the documents with spacy\(^4\), using the large models for each particular language. All words were lowercased. In the case of Newsela, we follow (Xu et al., 2015) and replace all named entities with a placeholder that represents the entity type.

During training, a teacher forcing strategy is followed half of the times. When teacher forcing is followed, the decoder is provided with the gold edit labels and target token; when not, the decoder at each time step is fed with the output produced in the previous edit label and target token. Default hyperparameters from EditNTS are used, and no hyperparameter tuning was performed: a batch size of 64, a hidden dimension of 200 and a learning rate of $10^{-3}$. We used Adam optimizer and a decay factor of $10^{-6}$. The models are trained during 50 epochs, and the model that obtained the best SARI score in the corresponding development split is selected and tested.
<table>
<thead>
<tr>
<th>Version</th>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orig. (EN)</td>
<td>He died on May 29, 1518 in Madrid, Spain and was buried in the church of San Benito d’Alcantara.</td>
</tr>
<tr>
<td>Ref.</td>
<td>He died on May 29, 1518 in Madrid, Spain. It was buried in the church of San Benito d’Alcantara.</td>
</tr>
<tr>
<td>Edit+synt</td>
<td>He died on May 29, 1518 in Madrid, Spain. he was buried in the church of San benito d’alcantara.</td>
</tr>
<tr>
<td>Edit</td>
<td>He died on May 29, 1518 in Madrid, Spain. buried in the church of san benito d’alcantara.</td>
</tr>
</tbody>
</table>

| Orig. (ES) | Los especialistas advierten que el asma se agrava en otoño. Specialists warn that asthma worsens in autumn.                           |
| Ref.       | Los especialistas advierten que el asma es peor en otoño. Specialists warn that asthma is worse in autumn.                        |
| Edit+synt  | los especialistas dicen de que el asma se reduce en otoño. Specialists warn that asthma is reduced in autumn.                   |
| Edit       | ma la cosa piú interessante non è questa. But this is not the most interesting thing. Ma non è questa la cosa piú grave.       |
|            | ma la cosa non può essere questa. But the thing cannot be this.                                                                                 |

| Orig. (IT) | Ma la cosa piú interessante non è questa. But this is not the most interesting thing. Ma non è questa la cosa piú grave.       |
| Ref.       | Ma non è questa la cosa piú grave. But this is not the most difficult thing.                                                                      |
| Edit+synt  | ma la cosa non può essere questa. But the thing cannot be this.                                                                                 |
| Edit       | ma la cosa piú importante non è. But the most important thing is not.                                                                             |

Table 5: Example of a sentence from Turkcorpus, Newsela-ES and the Italian corpus.

Regarding evaluation, the following evaluation metrics are used:

- **SARI** is a common evaluation metric for ATS systems (Xu et al., 2016) that measures the number of ngrams that have been added/removed/kept by the simplification system.

- **BLUE**. Following usual practice, we also include the BLUE score (Papineni et al., 2002) between the complex and simple sentences. Although BLUE has been criticized as a measure to evaluate simplification systems (Sulem et al., 2018; Dong et al., 2019; Martin et al., 2020b), we use it here for completeness.

We compute the evaluation metrics using the EASSE package (Alva-Manchego et al., 2019). We do not report the readability score Flesch–Kincaid Grade Level (FKLG) (Kincaid et al., 1975) because it is a language dependent metric which is only valid for English.

### 4.3 Baselines

We use the Identity baseline that simply copies the complex sentence. Apart from this, we compare our syntax-aware system against the state-of-the-art on each language. For English, we consider the deep reinforcement based neural system DressLS (Zhang and Lapata, 2017a), the transformer based model DMASS-DCSS (Zhao et al., 2018) and BART+ACCESS (Martin et al., 2020b), which also includes special tags to perform controllable text generation. There are fewer systems to compare against in the Spanish and Italian datasets, as, like many other areas in NLP, ATS systems have been developed mostly for English. For Newsela-es and the Italian dataset, we compare ourselves Neural TS (Palmero Aprosio et al., 2019), an MT system based on an attention encoder-decoder model. Finally, for Simplext we include the unsupervised system in Martin et al. (2020b). In the datasets for English we also report the results of EditNTS (Dong et al., 2019) with no syntax.

### 5 Results

Table 4 shows the results of our syntax aware system (dubbed Edit+synt in the tables), and compares them with the best performing systems on the different datasets. We see that the syntax aware system obtains very good results overall, and improves state of the art SARI results in four out of five datasets. This is a remarkable result that stresses the importance of syntax in text simplification. The table shows that datasets with smaller training data are most benefited from our approach, and suggests that the combination of edit operations and syntactic information is able to generalize in low training data regimes.

It is worth noting that the results obtained by us when running EditNTS without syntax are different to those reported in (Dong et al., 2019), and that the gap is specially large in the Wikilarge/Turkcorpus dataset\(^5\). We attribute this difference to the fact that the reported results in the original paper are obtained using the model that performed best in the test split, whereas we performed model selection according to the development dataset (c.f Section 4.2). There is also a slight difference in the Newsela dataset, which we attribute to the use of different evaluation scripts.

\(^5\)The latter results are marked with † in the tables.
Table 6: Main results. No syntax stands for the original system, whereas Syntax uses syntactic information. The ∆ measures the difference between both systems.

<table>
<thead>
<tr>
<th></th>
<th>No syntax</th>
<th>Syntax</th>
<th>∆SARI</th>
<th>∆BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wikilarge/TurkCorpus</td>
<td>36.75</td>
<td>36.97</td>
<td>0.22</td>
<td>2.36</td>
</tr>
<tr>
<td>Newsela</td>
<td>33.62</td>
<td>38.08</td>
<td>4.46</td>
<td>-1.85</td>
</tr>
<tr>
<td>Simplext</td>
<td>36.52</td>
<td>39.48</td>
<td>2.96</td>
<td>0.2</td>
</tr>
<tr>
<td>Newsela-es</td>
<td>34.50</td>
<td>35.03</td>
<td>0.53</td>
<td>-1.94</td>
</tr>
<tr>
<td>Italian</td>
<td>51.95</td>
<td>52.25</td>
<td>0.3</td>
<td>0.11</td>
</tr>
</tbody>
</table>

6 Analysis

In this section we perform an analysis of the results. We start by analyzing examples of the outputs, followed by an ablation study. Finally, we analyze the effect of the sentence complexity in the simplification process.

6.1 Output analysis

In Table 5 we present three sentences, one for each language that we explain next. In the English sentence (from Wikilarge), we see that both systems have been able to carry out a sentence splitting, but in the second sentence of the edit system the subject (he) is missing, which makes the ungrammatical (note that in the reference sentence, the pronoun is not correct). In the Italian sentence (from PaCCSS-it,SIMPITIKI, Terence-Teacher), the edit system has deleted the subject and it is ungrammatical (it could be grammatical with a different word ordering). The sentence created by Edit+synt has changed the modality of the verb (è ‘it is’-> può essere ‘it can be’)) and deleted the attribute più interessante ‘more interesting’, and this affects the meaning of the sentence. All in all, our analysis suggests that there is still a large room for improvement in non-English simplification. For example, the Spanish example in Table 5 is not correct at grammatical level: there is a dequesismo, which is the misuse of the preposition de in front of the conjunction que when the preposition is not required like in this case by the verb decir. This is not grammatically correct but it can be understood without problems. Moreover, the dequesismo is a common mistake by many speakers, and it would be interesting to check the corpus to find out if there are non standard grammatical variations or misuses. However, the main problem of the sentence is related to lexical simplification. The verb se agrava (it worsens) has been replaced with se reduce (it reduces). This is a wrong simplification, but if the sentence was not comprehensible at grammar level, the meaning preservation and simplicity cannot be correctly evaluated.

6.2 Ablation study

Table 6 shows the results of the system using syntactic dependencies or not. We see that, in general, syntactic dependencies are helpful and lead to an improvement in SARI on all datasets. The gain in SARI is particularly large in the Newsela and Simplext datasets, which are the datasets with highest average sentence depths. These results suggest that syntactic information is particularly helpful when simplifying complex sentences. This trend does not hold if we compare the SARI gain with the average sentence length. We analyze this correlation further in the next section. Regarding BLEU, the table shows mixed results, with gains in all datasets except in Newsela and Newsela-es.

6.3 The effect of the sentence complexity

The results in the ablation study indicate a correlation between the sentence complexity and the performance gain obtained when using syntactic information, and now we analyze this correlation further. Figure 2 shows an analysis of length and depth per sentence that helps understanding this relationship. The x axes in the figures correspond to the sentence depth (left) and length (right), and the y axes show the average SARI gain, that is, the average of the differences between the Syntax and No syntax scores for all sentences with one particular depth or length. While the results vary among datasets, the left graph shows a general tendency where the gain of using syntax is greater on the deepest sentences. That is, sentences that have complex dependency trees are better simplified, according to SARI, when using syntactic information.

The graphs have been smoothed using the Exponential Moving Average (EMA) technique with a smoothness factor of 0.75 to flatten the peaks. While the smoothing process removes information from the graph, the loss is outweighed by the improved visibility.
This gain is particularly high on the Newsela, WikiLarge/TurkCorpus and Simplext datasets.

The right graph in Figure 2 shows no overall correlation with the sentence length and the SARI gain. This result suggests that the length of a sentence per-se is not a valid indicator of the syntactic complexity of the sentences.

7 Conclusion and Future Work

In this paper we show that syntactic information is a valid source of information for edit-based text simplification systems. We have presented a system that extends a well known edit-based system with explicit syntactic information derived from dependency trees, by virtue of graph convolutional networks. The results show that the dependency information is useful, obtaining state of the art results on four out of five datasets in different languages. Further analysis show that the syntactic information is always beneficial (sometimes by a large margin), and that the improvement is often correlated with the depth of the dependency tree.

In the future we want to analyze the inclusion of dependency syntax information into transformer based seq2seq systems. In particular, we want to analyze whether explicitly modeling syntactic information is still a valid approach when the transformer based ATS system is initialized with large language models such as BART (Lewis et al., 2020).

Acknowledgments

This work was supported by the Basque Government (excellence research group IT1343-19 and DeepText project KK-2020/00088), project BigKnowledge (Ayudas Fundación BBVA a equipos de investigación científica 2018) and the Spanish MINECO (project DeepReading RTI2018-096846-B-C21).

References


337

On Generating Fact-Infused Question Variations

Arthur Deschamps∗
ByteDance Ltd.
Singapore
arthur.deschamps1208@gmail.com

Sujatha Das Gollapalli; See-Kiong Ng
Institute of Data Science
National University of Singapore
{idssdg,seekiong}@nus.edu.sg

Abstract

To fully model human-like ability to ask questions, automatic question generation (QG) models must be able to produce multiple expressions of the same question with different levels of detail. Unfortunately, existing datasets available for learning QG do not include paraphrases or question variations affecting a model’s ability to learn this capability. We present FIRS, a dataset containing human-generated fact-infused rewrites of questions from the widely-used SQuAD dataset to address this limitation. Questions in FIRS were obtained by combining a given question with facts of entities referenced in the question. We study a double encoder-decoder model, Fact-Infused Question Generator (FIQG), for learning to generate fact-infused questions from a given question. Experimental results show that FIQG effectively incorporates information from facts to add more detail to a given question. To the best of our knowledge, ours is the first study to present fact-infusion as a novel form of question paraphrasing.

1 Introduction

Recently, automatic Question Generation (QG) is being addressed for generating natural language questions for a given input text passage. Viewed as the reverse of the well-studied question answering task (QA), QG has been applied in education and tutoring (Heilman and Smith, 2010; Lindberg et al., 2013), dialog systems and chatbots (Shum et al., 2018), as well as for improving QA systems (Duan et al., 2017; Tang et al., 2018).

Various deep learning models are being rapidly developed for QG (Talmor and Berant, 2018; Kim et al., 2019; Tuan et al., 2020; Pan et al., 2020; Su et al., 2020; Wang et al., 2020a). However, it is only recently that QG studies have started focusing on an important aspect of the human question generation process known as paraphrasing, or the ability to ask questions in diverse ways all expressing the same intent (Harrison and Walker, 2018; Wang et al., 2020b).

Paraphrasing ability has been identified as a necessary aspect of learning human-like language generation (Shum et al., 2018; Huang et al., 2020) and was previously studied in context of community QA (Liang et al., 2016; Kunneman et al., 2019; Hosking and Lapata, 2021). These works address the identification of synonymous and syntactic question variations such as (“What’s the weight of an elephant in kg?”; “How heavy is an elephant?”). In addition to synonymous variations, human beings are also adept at generating questions expressing the same intent with varying level of details. For example, consider a QA pair from the SQuAD dataset1 shown in Table 1. SQuAD is one of the widely-used datasets for training QG models and contains about 100K training instances made up of an answer context, the answer string, and a “correct” question (Rajpurkar et al., 2016). We show in Table 1, fact-infused rewrites (or alternatively, variations) for the SQuAD question: “In what year did IBM get its name?”. Except question 2 which is a synonymous variation, the other variations include additional details of the entity “IBM” obtained from Google’s Entity Search API.2

We argue that question variations that include more detail can provide a form of query expansion and are likely to benefit downstream applications. Indeed, it has been observed that content words and named-entities referenced in the question improve the answerability of a question (Nema and Khapra, 2018) and result in improved QA and reading comprehension performance through the addition of

---

1https://rajpurkar.github.io/SQuAD-explorer/
2https://developers.google.com/knowledge-graph
In what year did IBM get its name?

The company originated in 1911 as the Computing-Tabulating-Recording Company (CTR) through the consolidation of The Tabulating Machine Company, the International Time Recording Company, the Computing Scale Company and the Bundy Manufacturing Company. CTR was renamed “International Business Machines” in 1924, a name which Thomas J. Watson first used for a CTR Canadian subsidiary. The initialism IBM followed. Securities analysts nicknamed the company Big Blue for its size and common use of the color in products, packaging and its logo.

Table 1: Example from the FIRS dataset.

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>In what year did IBM get its name?</td>
<td>1911</td>
</tr>
<tr>
<td>In what year did International Business Machines Corporation get its name?</td>
<td>1911</td>
</tr>
<tr>
<td>In what year did multinational technology company IBM get its name?</td>
<td>1911</td>
</tr>
<tr>
<td>In what year did American company IBM get its name?</td>
<td>1911</td>
</tr>
</tbody>
</table>

341
2 Related Work

Models for question generation and question answering are being rapidly developed in current NLP research. We refer our reader to a survey by Pan et al. (2019) for an overview on challenges, existing approaches, as well as evaluation metrics for QG. Several QG models use LSTM-based encoder-decoder setups with attention and copy mechanisms (Zhou et al., 2018; Duan et al., 2017; Zhao et al., 2018; Kim et al., 2019). Recent works are focused on improving QG performance by incorporating external knowledge, semantic information, and reinforcement learning into this basic architecture (Nema et al., 2019; Pan et al., 2020; Wang et al., 2020a; Majumder et al., 2021). Other state-of-the-art QG frameworks include variational autoencoders, graph convolutional networks and transformers (Lee et al., 2020; Su et al., 2020; Kriangchaivech and Wangperawong, 2019).

Paraphrase generation is a related task for identifying semantically similar texts in applications such as retrieval and question answering, query re-formulation and dialog system applications (Liang et al., 2016; Zhao and Wang, 2010). Similar to QG, seq2seq models and encoder-decoder architectures are common in paraphrase generation works (Gupta et al., 2018) but other approaches for paraphrase generation include variational autoencoders and translation models (Wang et al., 2019; Li et al., 2018; Hosking and Lapata, 2021).

3 FIRS Dataset Creation

As highlighted in Section 1, existing datasets for QA/QG and paraphrase generation do not include question variations with details. To fill this gap, we collected a new dataset by integrating the questions in the widely-used Stanford Question Answering Dataset (SQuAD) with relevant facts obtained from Google’s Entity Search API as follows:

**Collecting candidate question-entity pairs:** We selected from SQuAD, questions that refer to named entities in either the (1) question or (2) answer texts. In Table 1, we showed an example where the relevant entity *IBM* is mentioned in the question text. For the second case, consider a question from the SQuAD dataset from a passage on *Computer Security*, namely, “What is the source of the quote?” with the corresponding answer string “Reuters”. Nowhere in the SQuAD answer passage for this question is a mention of what “Reuters” is but using its entity description from Google, example paraphrases created by our crowdworkers for this question include “What news agency is the source of the quote?” and “Which international news organization is the source of the quote?”. This example highlights how a vague “What is” question can be expanded through the addition of the answer type (“news agency”) detail.

We obtained the subset of 25,316 questions from the 100K questions in SQUAD which reference ‘tangible’ named-entity types such as people, places, and organizations. Entity types referring to concepts such as “quantity, percent etc” are not supported in currently-available knowledge resources. For example, for the *IBM* question in Table 1, it is difficult to obtain focused knowledge pertaining to the answer “1924” (of type “date”).

<table>
<thead>
<tr>
<th>Entity Name</th>
<th>IBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>‘Corporation’, ‘Thing’, ‘Organization’</td>
</tr>
<tr>
<td>Description</td>
<td>Computer hardware company</td>
</tr>
<tr>
<td>Detailed description</td>
<td>International Business Machines Corporation is an American multinational technology company headquartered in Armonk, New York, with operations in over 170 countries.</td>
</tr>
</tbody>
</table>

Table 2: Search Result for “IBM” on the Entity Search API

**Obtaining Entity Descriptions:** We performed entity searches through the Google Knowledge Graph Entity Search API (GES) using the entity names as query strings. Next, entity-type match rules and text similarity thresholds were applied based on the source SQuAD passage to identify the correct entity description from the search results. We were able to obtain descriptions for 62,473 entities referenced in SQuAD questions using the above process. Based on crowdsourced annotations (described next), the precision of our search and filtering is \( \sim 97\% \). An example search result from GES along with its different fields is shown for the query “IBM” in Table 2.

We note that compared to other resources such as DBpedia (Lehmann et al., 2015) and YAGO (Hofart et al., 2013), the coverage of entities and facts is several scales higher in GES. After manually examining hundreds of results, we found GES to be consistently superior and accurate for our purpose. The “detailed description” fields were used by our crowdworkers while creating the question paraphrases. A limitation however is that there is no official documentation on the resources and algorithms employed in GES and neither is the full-

---

6[https://en.wikipedia.org/wiki/Knowledge_Graph](https://en.wikipedia.org/wiki/Knowledge_Graph)
type hierarchy information directly available given its proprietary nature.7

Creating ground truth paraphrases: We randomly sampled a subset of about 1600 (question, entity) pairs collected from Steps 1 and 2 for obtaining human-generated question variations. We set up our task through the crowdsourcing platform Amazon Mechanical Turk (AMT) following similar dataset collection efforts (Rajpurkar et al., 2016; Yang et al., 2018; Harrison and Walker, 2018). Each question, along with the entity descriptions was examined by three crowdworkers. The answer passage with the answer highlighted was also provided for the workers to identify cases where the entity is not relevant.

We required the crowdworkers to have greater than 95% HIT approval rate, a minimum of 10,000 HITs, and be located in the United States. The workers were instructed to “Rewrite the original question in more details using information from the provided knowledge” and to “Ensure that the intent of the original question remains the same.” Several examples of good and bad rewrites along with detailed explanations were included as guidance. At least one and up to three different re-writings were collected for each question per crowdworker.

After pooling the results of the AMT task, filtering out duplicates and variations that do not include any word from the extra knowledge (such as rewrite#2 for IBM in Table 1), our dataset has an average of four fact-infused variations for each question and is summarized in Table 3. We refer to our dataset as FIRS for Fact-Infused Rewrites of SQuAD questions.

3.1 Analysis of FIRS

We analyzed the question rewrites in FIRS along two dimensions, namely, (i) Diversity and (ii) Details. That is, a fact-infused rewrite should retain the semantics of the base question (original question from SQuAD) in terms of its intent but have other words that add extra details of relevant entities. To characterize this aspect, we employ the Simple Approximate Bigram Kernel (SBAK) similarity to measure the pairwise similarity between two sentences. Dependency-tree based similarity measures that account for partial matches and type of dependency edges are known to better represent semantic similarity between two sentences compared to bag-of-words similarity functions (Ambati, 2008; Özateş et al., 2016).

In Table 4, the average values of pairwise similarities of the fact-infused question variations with each other are shown in the “Intra-Set” column and with the base question are shown in the third column. The high SBAK similarity is indicative of semantic or intent similarity between the base question and the variations. However, the Jaccard overlap scores between the word sets (computed without stopwords) is lower due to the additional words present in the rewrites.8

The percentage spread of the parts-of-speech tags for the words added in rewritten questions are shown in Table 4. Not surprisingly, about 37% of the newly-added words are proper nouns or nouns whereas about 33% of words refer to adpositions, adjectives, and determiners that are often assigned to words surrounding noun phrases.9 These assignments indicate that the extra words added in rewrites are often content words and therefore, can be expected to improve the answerability of questions (Nema and Khapra, 2018).

Additional Notes on Data Collection: We performed the following checks to meet the ethics, quality, and reliability considerations for our collected questions. As part of the AMT data collection process, the anonymity and privacy of the crowdworkers is already ensured. Furthermore, the settings for the HIT approval rates, and location of

---

7Further details of the search result filtering processes are provided in the Appendix (See Footnote 5).

8The formulae from Özateş et al., 2016 are included in the Appendix for reference.

9https://universaldependencies.org
the worker, described previously are set similar to previous QA/QG data collection efforts to ensure the English language skills of the data annotators. A remuneration of $0.20 per assignment was paid to each worker. A total of 75 workers helped in creating our dataset, with about 47% of the workers labeling less than 5 questions each.

We ensure quality of the collected question variations, by employing a set of rules based on the a) similarity with the original question, (b) similarity within the collected paraphrases, (c) presence of the answer token, and (d) length of the rewritten question versus the original question. About 3% of the collected data was filtered out with the above rules. Finally, at every step of data collection, we performed checks manually on random subsets of the collected data to ensure the reliability of the named-entity taggers, the entity descriptions filtered from GES results as well as the quality of the rewrites produced by the workers.

4 Generating Fact-Infused Questions

Definition: Given two input sequences of \(|n|\) base question words, \(Q^b = q_1, q_2 \ldots q_n\), and \(|m|\) words of a fact related to an entity, \(F = f_1, f_2 \ldots f_m\), the objective of fact-infused question generation is to generate an output sequence of \(|k|\) words, \(Q^p = p_1, p_2 \ldots p_k\), such that \(Q^p\) is a fact-infused rewrite of \(Q^b\). That is, \(Q^p\) includes specific details from \(F\) while maintaining the intent of \(Q^b\) and our goal is to find \(Q^p\) that maximizes the conditional likelihood:

\[
Q^p = \arg\max_{Q} \mathcal{P}(Q|Q^b, F)
\]

We follow standard question generation approaches and adopt an attention-based encoder-decoder architecture for estimating the probability function, \(\mathcal{P}\) (Bahdanau et al., 2015; Sutskever et al., 2014; Kim et al., 2019). The main components of our Fact-Infused Question Generator Network (FIQG) are depicted in the schematic diagram in Figure 1 and summarized below:

**Fact and Question Encoders**: We use separate encoders for representing the question and fact sequences. The encoders are one-layer bidirectional LSTMs that extract contextual features from the input question (or alternatively, fact) and represent them as hidden states of the forward and backward LSTMs. Let \(E^i(Q^b)\) and \(E^i(F)\) represent the feature-rich embeddings of our base question and fact, respectively (Zhou et al., 2018). Then,

\[
E^i(Q^b) = [E^i(q_1), \ldots, E^i(q_n)]
\]

\[
E^i(F) = [E^i(f_1), \ldots, E^i(f_m)]
\]

where \(E^i(w)\) refers to the input feature embedding for word \(w\). Using LSTM notations, the hidden state for the fact encoder is therefore given by

\[
o_t^F = [\overrightarrow{h}_t^F; \overleftarrow{h}_t^F]
\]

where \(\overrightarrow{h}_t^F\) and \(\overleftarrow{h}_t^F\) are the hidden vectors of the forward and backward LSTMs, respectively, at time \(t\) and \(;\) represents the concatenation operator. The hidden state for the question encoder can be similarly represented as

\[
o_t^Q = [\overrightarrow{h}_t^Q; \overleftarrow{h}_t^Q]
\]

Next, applying the attention mechanism (Bahdanau et al., 2015) for the question encoders over its hidden states, the attention weighted sum of the contextualized question can be written as

\[
g_t = \sum_{i=1}^{n} \alpha_{ti} o_t^{Q^b}
\]

\[
\alpha_{ti} = \frac{\exp(a_{ti})}{\sum_{k=1}^{n} \exp(a_{tk})}
\]

\[
a_{ti} = f(s_{t-1}, o_t^Q)
\]

where \(\alpha_{ti}\) represent the attention weights with parameters \(a_{ti}\) such that \(Y_t^{Q^b} = \{\alpha_{ti}\}_{i=1}^{n}\) is a probability distribution over the question words. The values of \(a_{ti}\) depend on the hidden state of the decoder at the previous timestep \(s_{t-1}\), and the hidden state of the question encoder:

\[
f(s_{t-1}, o_t^Q) = v_E^\top \tanh(WE[s_{t-1}; o_t^Q])
\]

![Figure 1: Fact-Infused Question Generator Network.](image-url)
where $v_E$ and $W_E$ are learnable parameters.

The attention weights and vectors for the fact encoder are calculated similarly. We use $\gamma_{it}$ to refer to the parameterized and normalized attention weights for the fact encoder and $Y^F_t = \{\gamma_{jt}\}_{j=1}^m$ is a probability distribution over the fact words. The context vector for the fact can be written as:

$$m_t = \sum_{i=1}^{m} \gamma_{it} o^F_{it}$$

**Decoder:** The decoder takes the hidden states from the question and fact encoders to generate the paraphrased sequence of words. Our decoder is a uni-directional LSTM network whose state and context vectors are represented by $s_t$ and $i_t$, respectively, such that:

- $i_t = [E^d(p_{t-1}); m_{t-1}; g_{t-1}]$
- $s_t = LSTM(i_t, s_{t-1})$
- $s_0 = \overrightarrow{R}_1^0$

Here $E^d$ refers to the embedding from the decoder for the paraphrase word, $p_{t-1}$. The current context and decoder state vectors are combined with the attention vector from the question encoder to obtain the readout state and subsequently the generative probability distribution over the vocabulary using a maxout layer (Goodfellow et al., 2013):

$$r_t = W_r s_t + U_r i_t + V_r g_{t-1}$$
$$Y^V_t = \text{softmax}(W_y \text{ maxout}(r_t))$$

The matrices $W_y$, $W_r$, $U_r$ and $V_r$ are all learned during training.

**Copy mechanism:** Recent works for QG handle rare words by employing a pointer network that enables both copying of the words from the input source (answer passage) and generation of words during the decoding process (Gulcehre et al., 2016; See et al., 2017). For question rewriting using facts, we extend this copy mechanism to enable copying from both the input fact words as well as the question words. The copy switch in our case is a softmax function given by

$$p = \text{softmax} \left( W_{copy} s_t + U_{copy} g_{t} + Z_{copy} m_t + b \right)$$

where the matrices $W_{copy} \in \mathbb{R}^{3 \times |s_t|}$, $U_{copy} \in \mathbb{R}^{3 \times |g_t|}$ and $Z_{copy} \in \mathbb{R}^{3 \times |m_t|}$ are learnable parameters and $b$ is the bias parameter.

During the decoding step, $p$ is sampled to (1) copy the words from the question, based on $Y^Q_t$, the normalized, attention weights from the question encoder, or, (2) copy words from the fact based on $Y^F_t$, the normalized, attention weights from the fact encoder, or (3) generate a new word, based on $Y^V_t$, the generative distribution on the vocabulary estimated during learning.

### 4.1 Baselines

Considering the novelty of our proposed task, we are limited in our choice of baselines for comparing with FIQG. However, we note that similar to our objective which involves the infusion of parts of an entity fact into a given base question along with possible re-writing of the “wh”-word (for example, “Where” to “Which <location>”), existing QG approaches involve the inclusion of parts of an answer passage into a generated question template using attention and copy mechanisms. Thus, a straightforward application of QG models for our task would involve retraining the model using input passages comprising of both the base question and the fact sentences.

We also highlight that existing paraphrase generation models operate on a source question and generate synonymous variations by substituting specific words with related words and syntactic variations by using other exemplar questions (Fu et al., 2019; Hosking and Lapata, 2021). Consequently, we find QG models more appropriate for fact-infusion and compare FIQG with the following state-of-the-art QG baselines:

1. **NQG**\(^{10}\) is one of the earliest neural seq2seq models proposed for QG using feature-rich input embeddings comprising of words, answer position, parts-of-speech, NER and case information (Zhou et al., 2018).

2. **SGDQG**\(^{11}\) is a recent model designed to generate complex questions that require reasoning on multiple pieces of information (for example, in the HotpotQA dataset). SGDQG uses semantic graph information constructed from NLP relations between words in the passages (Pan et al., 2020).

3. **GSAQG**\(^{12}\) uses maxout pointer mechanism

\(^{10}\)https://github.com/magic282/NQG

\(^{11}\)https://github.com/YuxiXie/SG-Deep-Question-Generation

\(^{12}\)https://github.com/seanie12/neural-question-generation
with gated self-attention network to handle long text inputs (Zhao et al., 2018).

4. RefNet$^{13}$ is a two decoder based model where a second decoder refines the output from the first decoder for generating more complete questions (Nema et al., 2019).

5. ASs2s$^{14}$ employs an answer-separated seq2seq approach along with a keyword-net and interrogative word identification to handle irrelevant words in generated questions (Kim et al., 2019).

We note that QG models are being widely investigated in current research and some recent innovative aspects in learning QG include the use of variational encoders, graph convolutional networks, and incorporation of global and semantic knowledge (Pan et al., 2020; Wang et al., 2020a; Su et al., 2020; Majumder et al., 2021). Keeping the novelty of our task and dataset in mind, we compare against state-of-the-art models that use components very similar to FIQG and defer the investigation of more recent QG research on FIRS for future.

5 Experiments and Results

Implementation: We implemented FIQG in Python.$^{15}$ The hidden state sizes for the two encoders and the decoder are set to 256, whereas the depth for the attention mechanism is set to 512. The readout size is 128 whereas vocabulary size is $\sim 20K$ words, and the target sequence length was set to 50. Dropout rates are set to 0.5 for the dense layers and 0.3 for the attention layers, respectively. A learning rate of 0.001 was used.

Feature-rich embeddings (Zhou et al., 2018) were used for input representations using word, parts-of-speech tags and indicator embeddings. Indicator features using the BIO representation are incorporated in QG models to indicate the answer span in a passage to focus the question around the answer. For our case, this aspect corresponds to the named-entity whose fact we are integrating into the question. However, to differentiate the two cases, namely, when the entity is part of the answer versus when the entity name is part of the question, we use an extended set of tags: \{BA, IA, BN, IN, O\}.

Fact Extraction: The entity descriptions obtained from Google are brief summaries comprising 1-3 long sentences. The crowdworkers however only use specific segments of these summaries or entity facts in their paraphrases. To model this aspect, we used MinIE$^{16}$ an unsupervised, domain-independent fact extraction tool on our entity descriptions and mapped a specific fact from the summary with each rewrite (Gashteovski et al., 2017). We provide an example in the Appendix (Footnote 5).

Evaluation: Following existing QG works, we use BLEU (Papineni et al., 2002), METEOR (Lavie and Denkowsk, 2009), and ROUGE-L (Lin, 2004) scores to characterize model performance. All three measures are based on calculating the $n$-gram overlap between human-generated “gold” references and machine-generated predictions.

All baseline models were set up using the configuration settings shared by the authors. As in existing QG studies, we uniformly use pretrained embeddings from GloVe$^{17}$ (Pennington et al., 2014) for word representations and tune all models using the BLEU$^-4$ score on the dev portion of the dataset. All experiments were performed on a single GPU on an Nvidia cluster and FIQG took approximately 2 hours to train.

5.1 Results and Observations

Fact-Infusion Performance: In Table 5, we summarize the performance of FIQG and the baseline models using the different evaluation measures. FIQG is able to significantly outperform all baselines on the test data. Even though the number of training instances available in FIRS is significantly smaller than datasets such as SQuAD, fact-infused rewriting can be expected to be easier than standard QG since it involves combining a fact with a base question along with potentially rewriting the $wh$-word in contrast with QG where models learn to generate questions for a given passage and an answer-span. Indeed on SQuAD, the state-of-the-art QG models obtain BLEU$^-4$ and METEOR scores about half of that obtained on FIRS by FIQG. As such, the BLEU scores of the modified QG baselines on FIRS are also reasonably high although we note that separately representing the question and fact sentences via the double encoder in FIQG results in superior performance over

$^{13}$https://github.com/PrekshaNema25/RefNet-QG
$^{14}$https://github.com/yanghoonkim/NQG_A5s2s
$^{15}$Python 3.7.7, NLTK 3.5, Stanza 1.0.1 libraries were used in feature extraction whereas the deep learning models were implemented in Tensorflow 2.3.0.

$^{16}$https://github.com/uma-pi1/minie
$^{17}$http://nlp.stanford.edu/data/glove.840B.300d.zip
the baselines. Indeed, statistically significant gains are seen on all evaluation measures except the METEOR score for which the performance is similar to that of RefNet.

Ablation Experiments: The results of our ablation experiments are also shown in Table 5. Not surprisingly, and as shown in other QG studies, initializing our word embeddings with pretrained embeddings results in improved question rewriting performance. Without initialization from GloVe vectors, we observe a significant drop in the scores. Similarly, indicator features are known to help QG by providing signals to the model on what parts of the passage the generation should be focused on. For our smaller sentences, excluding them yields a small drop in performance. Moreover, discriminating between the two cases (answer versus question entity) seems to help the model attain a minor improvement in performance over using a single set of indicators as shown in the ‘w/ Combined Indicator’ row of Table 5. Although as observed in Section 3.1, the extra “fact” words in rewrites are often nouns and words related to nouns, excluding POS tag information causes a small drop in the performance. Based on these results, we can attribute the overall performance of FIQG mostly to the network architecture coupled with appropriately initialized word embedding features.

Anecdotal observations: We show sample test predictions with FIQG in Table 6 for discussion. In the first example, a fact related to an entity mentioned in the question (“Martin Luther”) is being utilized whereas in the second example, the fact relates to “David Booth”, the answer entity. FIQG missed some words from the human-specified variation (“target”) in the first case and gets the tense wrong in the second example. However, we note that the fact extracted from the summary did not contain the extra initials whereas the tense is also specified incorrectly in the base question from SQuAD. Barring these minor aspects, the predictions are legitimate and complete and in the second example we also note the change in the wh-word.

![Table 5: Question Paraphrase Generation Results on FIRS](image)

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>NQG</td>
<td>0.431</td>
<td>0.318</td>
<td>0.247</td>
<td>0.195</td>
<td>0.222</td>
<td>0.484</td>
</tr>
<tr>
<td>SGDQG</td>
<td>0.524</td>
<td>0.374</td>
<td>0.278</td>
<td>0.209</td>
<td>0.233</td>
<td>0.482</td>
</tr>
<tr>
<td>RefNet</td>
<td>0.567</td>
<td>0.469</td>
<td>0.397</td>
<td>0.338</td>
<td>0.381</td>
<td>0.562</td>
</tr>
<tr>
<td>GSAQG</td>
<td>0.572</td>
<td>0.472</td>
<td>0.390</td>
<td>0.322</td>
<td>0.293</td>
<td>0.589</td>
</tr>
<tr>
<td>ASs2s</td>
<td>0.614</td>
<td>0.497</td>
<td>0.411</td>
<td>0.342</td>
<td>0.292</td>
<td>0.579</td>
</tr>
<tr>
<td>FIQG (Our Model)</td>
<td>0.729</td>
<td>0.623</td>
<td>0.547</td>
<td>0.486</td>
<td>0.382</td>
<td>0.686</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ablation Experiments</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>-GloVe</td>
<td>0.634</td>
<td>0.510</td>
<td>0.429</td>
<td>0.367</td>
<td>0.331</td>
<td>0.623</td>
</tr>
<tr>
<td>-Indicator Features</td>
<td>0.721</td>
<td>0.608</td>
<td>0.528</td>
<td>0.464</td>
<td>0.376</td>
<td>0.675</td>
</tr>
<tr>
<td>-POS Features</td>
<td>0.705</td>
<td>0.598</td>
<td>0.523</td>
<td>0.463</td>
<td>0.371</td>
<td>0.677</td>
</tr>
<tr>
<td>w/ Combined Indicator</td>
<td>0.717</td>
<td>0.608</td>
<td>0.531</td>
<td>0.469</td>
<td>0.376</td>
<td>0.676</td>
</tr>
</tbody>
</table>

347

6 Conclusions and Future Work

We presented FIRS, to the best of our knowledge, a first-of-its-kind dataset containing fact-infused vari-
ations of a subset of questions from SQuAD. We proposed a double encoder-decoder model FIQG, for learning to generate question variations through fact infusion. FIQG is able to significantly outperform extensions of standard QG models on FIRS.

In future, we would like to investigate the use of question variations on downstream tasks such as QA, reading comprehension, and interactive dialog (Tang et al., 2018; Ribeiro et al., 2019; Gao et al., 2020). Additionally, question variations available in FIRS can be used for learning diverse question generation, adversarial models for QA, and QG on multiple passages (Ren et al., 2018; Yang et al., 2018; Gan and Ng, 2019). We would like to explore these aspects as well as study novel learning methods such as variational auto-encoders and reinforcement learning for improving performance on the fact-infused question generation task (Misra et al., 2018; Li et al., 2018).

**Ethics Statement**

This research was conducted in conformance with the ACM Code of Ethics.

**Acknowledgements**

This research is supported by the National Research Foundation, Singapore under its Industry Alignment Fund – Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

**References**


Heung-yeung Shum, Xiao-dong He, and Di Li. 2018. From eliza to xiaoice: challenges and opportunities with social chatbots. In Frontiers of Information Technology & Electronic Engineering.


Event Prominence Extraction Combining a Knowledge-Based Syntactic Parser and a BERT Classifier for Dutch

Thierry Desot, Orphée De Clercq, and Veronique Hoste
LT3, Language and Translation Technology Team
Ghent University, Groot-Brittannielaan 45, 9000 Gent, Belgium
{thierry.desot, orphee.declercq, veronique.hoste}@ugent.be

Abstract
A core task in information extraction is event detection that identifies event triggers in sentences that are typically classified into event types. In this study an event is considered as the unit to measure diversity and similarity in news articles in the framework of a news recommendation system. Current typology-based event detection approaches fail to handle the variety of events expressed in real-world situations. To overcome this, we aim to perform event salience classification and explore whether a transformer model is capable of classifying new information into less and more general prominence classes. After comparing a Support Vector Machine (SVM) baseline and our transformer-based classifier performances on several event span formats, we conceived multi-word event spans as syntactic clauses. Those are fed into our prominence classifier which is fine-tuned on pre-trained Dutch BERT word embeddings. On top of that we outperform a pipeline of a Conditional Random Field (CRF) approach to event-trigger word detection and the BERT-based classifier. To the best of our knowledge we present the first event extraction approach that combines an expert-based syntactic parser with a transformer-based classifier:

- Input sentences are first pre-processed using a rule-based syntactic parser in order to generate smaller syntactic clauses as multi-word event spans.
- In a second phase, event prominence classification is applied in order to express whether it is a main or background event, using a classifier which is fine-tuned on pre-trained Dutch BERT word embeddings.

We also motivate the use of syntactic clauses as event spans, by comparing baseline and target classifier performances on other multi-word event span formats. On top of that we outperform a pipeline of a CRF event-trigger word detection approach

1 https://www.ugent.be/mict/en/research/NewsDNA is an interdisciplinary research project at Ghent University that aims to outline a news recommendation algorithm driven by diversity of topics and events that occur in unfiltered news streams.

1 Introduction
Recently, news publishers have shifted from newspapers to digital means which provide news readers easy access to a wide range of news information. However, the challenge is to find the right content that also corresponds to the user’s personal interests. Therefore, many of today’s major media and news websites offer automated news recommendation and personalization (Das et al., 2007; Odić et al., 2013; Moreira et al., 2019; Feng et al., 2020). News personalization paradigms define good news recommendations in terms of similarity to the user’s previous reading behaviour. Hence, news articles are recommended based on proximity to other articles the user has read (Liu et al., 2010; Adnan et al., 2014). However, this contrasts with the normative concept of journalism that stimulates diversity of topics and events in unfiltered news streams (Pariser, 2011; Joris et al., 2019). In this study we consider the news event as a means to model both diversity and similarity in news articles in the context of a news recommendation system.

We present an event extraction approach that will be integrated in a news recommender for Dutch¹. As current typology-based event detection fails to handle the variety of events in real-world situations we applied event prominence classification. This allows us to detect unrestricted news events and to overcome the sparsity of a small training data set. Our event extraction approach combines an expert-based syntactic parser with a transformer-based classifier:

- Input sentences are first pre-processed using a rule-based syntactic parser in order to generate smaller syntactic clauses as multi-word event spans.
- In a second phase, event prominence classification is applied in order to express whether it is a main or background event, using a classifier which is fine-tuned on pre-trained Dutch BERT word embeddings.

We also motivate the use of syntactic clauses as event spans, by comparing baseline and target classifier performances on other multi-word event span formats. On top of that we outperform a pipeline of a CRF event-trigger word detection approach

¹https://www.ugent.be/mict/en/research/NewsDNA is an interdisciplinary research project at Ghent University that aims to outline a news recommendation algorithm driven by diversity of topics and events that occur in unfiltered news streams.
and our BERT-based classifier. Furthermore, our approach is positioned with respect to the state of the art in Section 2 and is outlined in Section 3. An overview of the data set is given in Section 4. Section 5 presents the results of experiments on the held-out test set followed by a results analysis and discussion, conclusion and outlook on future work.

2 Related Work

Knowledge-based approaches are still frequently used for event extraction. Such methods are based on ontologies (Frasincar et al., 2009; Schouten et al., 2010; Arendarenko and Kakkonen, 2012) or rule-sets (Valenzuela-Escárcega et al., 2015) which represent expert knowledge. Information is mined from corpora based on lexical, syntactic (Hearst, 1992, 1998) and semantic patterns or frames (Cunningham, 2002a,b; Xie et al., 2013; Borsje et al., 2010; Hogenboom et al., 2013).

As the manual creation of rule-sets and ontologies is difficult and time-consuming, data-driven event extraction approaches made their entrance. The ACE (Automatic Context Extraction) annotation standards\(^2\), ERE (Entities, Relations, Events) annotation standards (Song et al., 2015; Aguilar et al., 2014) and TAC-KBP (Text Analysis Conference Knowledge Base Population)\(^3\) workshops and competitions stimulated the creation of data sets labeled with entities and events, e.g. the ACE 2005 corpus (Walker et al., 2006). As a consequence, supervised methods became predominant but initially concentrated on fixed event types using single-word event spans (Mitamura et al., 2015a). As compensation for small event spans, sentence or cross-sentential context information was used. In Ji and Grishman (2008) and Hong et al. (2011) events were extracted through cross-document and cross-sentence inference, respectively. Liao and Grishman (2011) improved event extraction performances adding topic classification information.

As feature engineering approaches emerged, a larger scope than one-word event spans was targeted. Hand-designed sets of lexical, semantic or syntactic features were extracted and fed into classifiers, allowing the model to take more context into account (Patwardhan and Riloff, 2009). Event extraction tasks are typically applied in a pipeline architecture where event trigger word identification, argument and event classification are conceived as separate tasks (Ahn, 2006). Other than a pipeline architecture, multi-task architectures perform several subtasks simultaneously to benefit from their interdependencies. In Li et al. (2013) events were extracted incorporating features that capture dependencies of multiple triggers and arguments. Luan et al. (2019) and Wadden et al. (2019) extracted events combined with named entity and argument role prediction.

However, the choice of features is a manual and elaborate process that requires extensive linguistic domain expertise. More recently deep neural networks superseded methods that show a strong dependency on feature resources, although the latter ones are still not definitely outperformed. Jacobs et al. (2018) and Nugent et al. (2017) used lexical, syntactic features, word2vec (Mikolov et al., 2013), glove (Pennington et al., 2014) and fastText (Bojanowski et al., 2017) word embeddings. Better performances were reported for an SVM classifier compared to a Recurrent Neural Network (RNN). In contrast, Nguyen and Grishman (2015) demonstrated that Convolutional Neural Networks (CNN) significantly outperformed feature-based methods on the ACE 2005 task.

Meanwhile, contextual language models have proven successful in a transformer architecture (Vaswani et al., 2017) that fully benefits from the attention mechanism. It has been integrated in a range of NLP tasks using pre-trained contextual BERT (Bidirectional Encoder Representations from Transformers) word embeddings (Devlin et al., 2018), predominantly for English. Mao and Liu (2019) report encouraging results for an event factuality classifier using BERT. Piskorski et al. (2020) report SVM event classifications with Term Frequency-Inverse Document Frequency (TF-IDF) that are outperformed by a fine-tuned BERT event classifier. The results of these studies inspired us to combine an expert-based syntactic parser with a BERT-based language model classifier for Dutch in order to extract multi-word events.

3 Method

3.1 Event Extraction for News Recommendation

In this study, an event is considered as the unit to measure proximity to other articles the user has read for news recommendation. It can be defined as the smallest extent of text that expresses its occurrence.
(Song et al., 2015), or a change of state at a particular place and time (Mitamura et al., 2015b), and is identified by a word or phrase called event trigger, nugget, event span or mention. Event mentions can be single-word event triggers that are usually (main) verbs, nouns, adjectives and adverbs. Multi-word event triggers can be continuous when the event span consists of consecutive tokens and even complete sentences, or discontinuous when its participants, or argument roles are also involved (Doddington et al., 2004). As they are more challenging to predict, we initially performed event classification on event spans with a fixed and short length, i.e. 5 token windows with a verbal head only. In a second phase we targetted longer events with a variable length, i.e. annotated events and syntactic clauses (Sections 5.1 and 5.2). The event extraction process in this study consists of automatically assigning an event prominence label to continuous multi-word event spans from a held-out test set. For the Dutch input document (translated in English) in Figure 1, the Main event is about a promotion campaign activity; the Background event provides background information about the Main event. Our hypothesis is that our target transformer classifier model is capable of categorizing new information into more general prominence classes, fine-tuned on pre-trained BERT word embeddings.

### 3.2 Syntactic Pre-Processing and Extraction of 5 Token Windows

Multi-word event spans, in this study defined as syntactic clauses as output from raw sentences processed by the Alpino syntactic parser, are fed into our baseline and target event prominence classifiers. The complete process is depicted in Figure 2.

The Alpino parser’s knowledge-based part consists of a rule-based head driven phrase structure grammar (HPSG) and lexicon (100,000 entries). The integrated part-of-speech (POS) tagger reduces lexical ambiguity. The resulting dependency parse trees are disambiguated with a maximum entropy component (Van der Beek et al., 2002; Van Noord et al., 2006; Smessaert and Augustinus, 2010). An F-score of 91.14% was measured for 1,400 manually annotated sentences from the Twente News corpus (Ordelman et al., 2007).

For our experiments we applied a set of rules on the parser output in order to split sentences in the test set into separate main and subclauses. Subclauses in sentence medial position were not considered, but only in sentence initial and final position. In this way, the syntactic structure of our pre-processed test sentences is more similar to the clauses in the training set. For the Dutch sentence⁴ in row 1 of Table 1, the labels ssub (subclause), begin and end position are used to extract the relative subclause from the syntactic parser output in row 3. As a preparatory step event classification was first performed on fixed event spans with a short length. To that end main head verbs in a 5 token window context were extracted from the annotated events in our data, also by applying rules on the syntactic parser output.

We compared our syntax-driven event extraction approach with a CRF⁵ (Lafferty et al., 2001) model to event detection as outlined in Colruyt et al. (under review), combined with our target classifier. For an input sequence of lexical, word shape and syntactic features, the CRF predicts a target sequence in IOB format. Tokens starting an event mention are labelled as B, tokens inside the mention as I, and tokens outside the mention are labeled as O.

<table>
<thead>
<tr>
<th>Raw input sentence</th>
<th>Soldiers zullen worden ingezet in de wijk Rocinha die zo’n 70.000 inwoners telt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin and end position of a subclause</td>
<td>&lt;begin=9 cat=ssub end=13&gt;</td>
</tr>
<tr>
<td>Extracted (relative) subclause</td>
<td>die zo’n 70.000 inwoners telt</td>
</tr>
</tbody>
</table>

Table 1: Subclause extracted from syntactic parser output

### 3.3 Baseline Classification Models

For a prominence classification of multi-word event spans, i.e. 5 token windows or syntactic clauses, into Main, Background and None event labels, an SVM classifier was trained as baseline model using the scikit-learn Python library. SVM performances were compared for Bag of Words (BOW) and TF-IDF count-based methods. Instead of deriving meaning from an entire corpus, word representations are constructed one sentence at a time, with a prediction-based method that predicts word identity given a sentence context. The model

---

⁴English translation: “Soldiers will be deployed in the Rocinha district, which includes about 70,000 inhabitants.”

learns that words occurring in similar sentence contexts are semantically related. This was applied by combining the SVM classifier with Dutch pre-trained word2vec word embeddings (Tulkens et al., 2016). The embeddings were pre-trained on the combined Dutch Roularta\textsuperscript{6}, Wikipedia\textsuperscript{7} and SoNaR corpora (Oostdijk et al., 2013) with a total of 54.8 million sentences and 803 million words.

### 3.4 Transformer-Based Target Classification Model

SVM baseline performances for event prominence classification were compared with a transformer-based (Section 2) classifier that relies entirely on the self-attention mechanism. It relates different positions of a single sequence in order to compute a representation of the sequence (Vaswani et al., 2017). For an input sequence $x = (x_1, ..., x_n)$ of $n$ elements, where $x_j \in \mathbb{R}^{d_z}$ each attention head in the self-attention sublayers calculates a sequence $z = (z_1, ..., z_n)$, where $z_i \in \mathbb{R}^{d_z}$. Each output element, $z_i$, is computed as weighted sum of linearly transformed input elements,

$$z_i = \sum_{j=1}^{n} \alpha_{ij} (x_j W^V)$$  \hspace{1cm} (1)

Each weight coefficient, $\alpha_{ij}$, is calculated with a softmax function,

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$  \hspace{1cm} (2)

and $e_{ij}$ is computed with a function comparing two input elements,

$$e_{ij} = (x_i W^Q)(x_j W^K)^T \frac{1}{\sqrt{d_z}}$$  \hspace{1cm} (3)

where $W^Q, W^K$ and $W^V \in \mathbb{R}^{d_z \times d_z}$ are parameter indices that are unique per layer and attention head. The attention function maps vectors of queries $W^Q$ and key-value pairs $W^K$, $W^V$ to an output (Shaw et al., 2018).

BERT are unsupervised deep bidirectional word embeddings (Devlin et al., 2018) pre-trained on large corpora in the target language. Frequently, a smaller dataset is used for fine-tuning for the target NLP task. A replication study and evaluation of BERT resulted in RoBERTa (Liu et al., 2019) that is trained on more data, bigger batches and longer sequences. Bidirectional pre-training is realized with a masked language model (MLM). The MLM randomly masks input tokens in order to predict the original vocabulary relying on its left and right context. In addition to the MLM next sentence prediction (NSP) jointly pre-trains text-pair representations.

A Dutch BERT model, BERTje (de Vries et al., 2019) has been pre-trained on a dataset of 2.4 billion tokens from Wikipedia, Twente News Corpus (Oordelman et al., 2007), and SoNaR-500 corpora (Oostdijk et al., 2013). RobBERT (Delobelle et al., 2020), a RoBERTa based and larger model has
Table 2: EventDNA corpus statistics

<table>
<thead>
<tr>
<th>Events</th>
<th>Entities</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td>4248 PER</td>
<td>6943 Vocabulary 13276</td>
</tr>
<tr>
<td>Backgr.</td>
<td>3154 LOC</td>
<td>5537 Tokens 90062</td>
</tr>
<tr>
<td>None</td>
<td>1824 ORG</td>
<td>4441 Sentences 6924</td>
</tr>
<tr>
<td>Total</td>
<td>9226 MISC</td>
<td>17411 Documents 1771</td>
</tr>
</tbody>
</table>

been pre-trained on 6.6 billion Dutch tokens from the OSCAR corpus (Suárez et al., 2019). Other than BERTje, RobBERT does not integrate NSP. Both models have an architecture of 12 transformer blocks (bidirectional layers) and 12 self-attention heads and a hidden size of 768.

4 Data

Our baseline and target event prominence classification models were trained on the EventDNA corpus. It comprises 1,771 Dutch news articles (Table 2, Documents), of which only the title and lead paragraph were kept, and is annotated with entities, news events and IPTC (International Press Telecommunications Council) Media Topic codes (Colruyt et al., under review). The annotation protocol was based on the ERE (Entities, Relations, Events) annotation standards (Song et al., 2015; Aguilar et al., 2014).

Entity spans can be assigned one out of four possible labels: person (PER), location (LOC), organization (ORG), and (MISC) for other entity values (Table 2, Entities). A sentence can comprise more than one event (with an average of 1.3 events per sentence). All relevant semantic information (with priority over syntactic information) is included in the event span that can contain entire, main or subclauses, or nominal expressions. Hence the event’s arguments can be included. An Event span is annotated with a prominence feature label: Main events bring new information and actually caused the reporter to write the article; Background events give context or background to the Main event; raw sentences without events are labeled as None events (Table 2, Events). Our motivation to apply prominence classification other than event type labeling is mainly driven by a prior analysis of the EventDNA corpus which revealed a high frequency (32%) of event types in a small data set (Table 2, Sentences) that cannot be classified into one of the event types specified in the EventDNA annotation protocol. Figure 1 presents an example of an event span labeled as Background event, preceded by a Main and None event. For more information about the EventDNA annotations we refer the reader to Colruyt et al. (2019).

For our experiments, both data sets with annotated events and 5 token windows with verbal head, extracted from the corpus, were randomized and split into 80% train, 10% development (DEV) and 10% held-out test data as shown in Table 3. The number of 5 token window instances in the training and test set is lower than the number of annotated events, as only events with a verbal head were extracted. Subsequently, performance comparisons between the models trained on those two data sets in Section 5.1 are not entirely fair. For that reason we provided a test set with only overlapping instances between the 5 token window instances and the annotated event instances for a fair comparison (Table 3, Annotated events2).

In order to verify the feasibility of our approach to classify events based on the test sentences, split into syntactic clauses, with the Alpino syntactic parser (Section 5.2), we counted the syntactic constituents in the training data annotated with events. Table 4 shows that the majority of the annotated events in the training set consist of a single verbal main-, subclause or infinitival construction. By splitting our test input sentences into syntactic clauses the syntactic structure of our pre-processed test sentences is more similar to the single verbal main-, subclause or infinitival construction (50.97%) and main clauses combined with other verbal constituents (13.57%) in the training set.

As the test sentences were split into syntactic clauses, the number of test instances (Syntactic clauses) in Table 3 exceeds the number of the original Raw test sentences. Hence, performance comparisons on the Raw sentences and Syntactic clauses for the syntax based event extraction experiments in Section 5.2 are not entirely fair. However, the test sets in Table 3, used for our experiments in section 5, are based on the same 10% held-out test data from the EventDNA corpus. We mapped the raw sentence and syntactic clause test set versions with the Annotated events in order to assign the event labels, and manually verified these. For raw sentences comprising several events, we randomly assigned one event prominence class. We also pro-

---

8https://iptc.org/standards/media-topics/
5 Experiments and Results

We trained and tested our SVM baseline event classifier and target BERT event classifier on 5 token windows and annotated events (Section 4). Then we fed the syntactic clauses from the syntactic parser into the baseline SVM and target BERT classifiers. Finally, we positioned our approach w.r.t. a pipeline of a CRF approach to event-trigger word detection and target prominence classifier (Section 5.2).

5.1 Event Extraction Based on 5 Token Windows and Gold-Standard Events

For training the SVM baseline event classification models (Section 3.3), parameters were optimized using the DEV set. The best results were obtained with an RBF kernel with cost $C = 20$, using the default scale value of the parameter $\gamma$, applying one-vs-rest classification. SVM performances are compared for BOW, TF-IDF, and pre-trained word2vec Dutch word embeddings. For fine-tuning the target BERTje and RobBERT prominence classifiers (Section 3.4), AdamW optimizer was used (Loshchilov and Hutter, 2017) with a learning rate of $1e{-5}$ and a batch size of 10 instances. The maximum sequence length is similar to 69 tokens, which is the maximum sequence token length of the annotated events in the training data. As we are interested in single sentence classification we added the special [CLS] (classification) token. Minimal loss was obtained after 3 epochs of training for BERTje and 4 epochs for RobBERT with a cross entropy loss function. Performances were evaluated using Recall (Rec.), Precision (Prec.) and F-score.

Surprisingly, the SVM baseline classifier with word2vec embeddings did not outperform the SVM TF-IDF and BOW models (Table 5). However, the study of Tulkens et al. (2016) also reported varying performances for the Dutch word2vec embeddings compared to BOW and TF-IDF. In general, better performances are exhibited for the models trained on the annotated events than for the 5 token windows. For both data sets the transformer models outperform the SVM classifiers with slightly superior performances for RobBERT on the 5 token windows and BERTje on the annotated events. For the latter model, Table 6 exhibits worst performances on the Background prominence class, compared to Main and None classes.

5.2 Syntax Based Event Extraction

As we defined our target multi-word event spans as syntactic clauses (Section 3.1), the raw sentences in the test set were pre-processed with the syntactic parser outlined in Section 3.2, before feeding the resulting clauses to the baseline SVM and target BERT classifiers as used in Section 5.1. Table 7 shows best performances for the BERTje classifier on syntactic clauses, that are very similar to syntactic clauses 2, the syntactic clauses that were aligned (Section 4) with the Raw sentences for a fair comparison.

We also compared our event extraction approach using the BERTje model that classifies multi-word event spans, conceived as syntactic clauses, with a pipeline consisting of a CRF for event-trigger word detection (Section 3.2) and our BERT-based classifier. The CRF model was trained for ten iterations on the annotated Main and Background events in the training set (Section 4) and tested on the raw sentences in the held-out test set (Table 3). Only

<table>
<thead>
<tr>
<th>Data set</th>
<th>Instances training set</th>
<th>Instances test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotated events</td>
<td>7362</td>
<td>934</td>
</tr>
<tr>
<td>Annotated events2</td>
<td>7362</td>
<td>780</td>
</tr>
<tr>
<td>5 token windows</td>
<td>6248</td>
<td>780</td>
</tr>
<tr>
<td>Raw sentences</td>
<td>-</td>
<td>904</td>
</tr>
<tr>
<td>Syntactic clauses</td>
<td>-</td>
<td>1030</td>
</tr>
<tr>
<td>Syntactic clauses2</td>
<td>-</td>
<td>904</td>
</tr>
</tbody>
</table>

Table 3: Training and test sets - annotated events, 5 token windows, raw sentences and syntactic clauses

<table>
<thead>
<tr>
<th>Single syntactic constituent</th>
<th>Annotated events Train set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-verbal:</td>
<td></td>
</tr>
<tr>
<td>Noun Phrase</td>
<td>35.44</td>
</tr>
<tr>
<td>Verbal:</td>
<td></td>
</tr>
<tr>
<td>Infinitival construction</td>
<td>1.84</td>
</tr>
<tr>
<td>Main clause</td>
<td>44.93</td>
</tr>
<tr>
<td>Subclause</td>
<td>4.20</td>
</tr>
<tr>
<td>Main clause + verbal const.</td>
<td>13.57</td>
</tr>
</tbody>
</table>

Table 4: Syntactic constituents in EventDNA training data
the resulting detected (67% F1-score) Main and Background events, without the None events, were fed into the transformer classifier. Table 8 exhibits significantly poorer prominence classification results on the CRF detected events compared to classification on the syntactic clauses (also without the None events).

6 Results Analysis and Discussion

Analysis of BERTje attention heatmaps indicated the feasibility of our event extraction approach combining a syntactic parser and a BERT classifier. The sentence “Then an adviser to the president was convicted because he lied” (Figure 3 - left) consists of a main clause “Then an adviser to the president was convicted” (middle) with a main event, and a subclause “because he lied” (right). Figure 3 (left) shows that most attention in the raw sentence is erroneously attributed to the past participle in the subclause, “gelogen” (lied). After splitting the sentence in its main and subclause, most attention is now correctly attributed to the verbs in the Main (middle) and Background (right) event. Although the BERTje classifier performances on the syntactic clauses are better, compared to the CRF detected events (Table 8), classification performances are still poorer compared to classification on the test set with annotated events (Table 5). As the training data has been annotated taking into account semantic information, with priority over syntactic information, the boundaries of the syntactic clauses generated by the Alpino parser, are frequently different from the boundaries of the annotated events which results in poorer performances. On top of that 35.44% of the EventDNA training data consists of non-verbal constituents (Table 4). These are mainly news article titles, but also noun phrases as part of a main clause that have been annotated as separate events. However, our rule-set on top of the syntactic parser, splits raw test sentences into separate main and subclauses (Section 3.2), but does not isolate nominal constituents. This also partially explains poorer performances on the syntactic clauses compared to the annotated test events. A possible solution for this bottleneck is combining the rule-set on top of the syntactic parser, with the BERTje self-attention mechanism. Tokens in the syntactic clause to which the highest attention values are attributed can be extracted, e.g. nominal constituents as part of a clause.

The transformer models outperform the SVM (Section 5) and benefit from the structure of language that is taught during pre-training. Certain self-attention heads exhibit linguistic notions of syntax and coreference. In line with the studies of Vig (2019), Vig et al. (2019) and Clark et al. (2019), coreference relations are situated in the middle and deeper layers of the self-attention blocks as depicted in Figure 4. For the sentence “She survived the bullet to her head”, coreference between the Dutch personal pronoun ze (she), on the right, and the possessive pronoun, on the left, haar (her) is depicted as connecting lines. Darker colors represent higher attention weights. In general

<table>
<thead>
<tr>
<th>Model</th>
<th>5 token windows</th>
<th>Annotated events</th>
<th>Annotated events 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (BOW)</td>
<td>56.56</td>
<td>58.38</td>
<td>56.62</td>
</tr>
<tr>
<td>SVM (TF-IDF)</td>
<td>56.61</td>
<td>58.00</td>
<td>56.63</td>
</tr>
<tr>
<td>SVM (Word2vec)</td>
<td>52.96</td>
<td>53.64</td>
<td>53.24</td>
</tr>
<tr>
<td>BERTje</td>
<td>57.18</td>
<td>58.07</td>
<td>57.29</td>
</tr>
<tr>
<td>RobBERT</td>
<td>57.89</td>
<td>58.46</td>
<td>58.13</td>
</tr>
</tbody>
</table>

Table 5: SVM, BERTje and RobBERT event classification performances (%), trained and tested on 5 token windows and annotated events

<table>
<thead>
<tr>
<th>Test set events (%)</th>
<th>Annotated events</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec.</td>
</tr>
<tr>
<td>Backg. (34.90)</td>
<td>68.01</td>
</tr>
<tr>
<td>Main (45.08)</td>
<td>71.24</td>
</tr>
<tr>
<td>None (20.02)</td>
<td>75.43</td>
</tr>
</tbody>
</table>

Table 6: BERTje classification performances (%) on annotated events per prominence class

Original Dutch sentence: “Ze overleefde de kogel door haar hoofd”
In spite of the advantages of using the transformer model, minimal loss was already obtained after only 3 epochs of training for BERTje (Section 5.1). The BERTje model pre-trained on a large corpus, allows a small dataset being used for fine-tuning on the event prominence classification task. However, applying data augmentation on the small NewsDNA dataset might increase training time during fine-tuning. Although the pre-trained BERTje model is large (2.4 billion tokens), it contains other data than news corpora, whereas our training set consists entirely of news. This raises the question whether it is not better to use a domain-specific pre-trained model consisting entirely of news corpora.

A bottleneck of classifying prominence labels only based on the sentence level, is the lack of context information. This has an impact mainly on the Background prominence class (Table 6). Semantic and syntactic information cues within a sentence can in some cases be sufficient to correctly predict a Background class. E.g. the conjunction “when” in “when she tried to convince the shooter” introduces a subclause with a noun “shooter”, which refers to a shooting or killing Main event outside the subclause that contains a Background event “convince”. However, frequently more context information is necessary in order to correctly pre-
dict the Background prominence label. As a next research step, for fine-tuning the transformer model, extra separator tokens [SEP] with previous and/or next annotated events can be inserted to the current training instances. This can provide the model more context to improve Background prominence class predictions. Furthermore, instead of using event prominence classes, more generalized event types can be generated, by mapping the original more specific event types in the NewsDNA data to broader event classes. This would decrease the need for more context information. However, the latter approach might not offer the complete solution to handle the variety of events expressed in real-world situations.

7 Conclusion and Future Work

This study shows that an event extraction approach of an expert-based syntactic parser in combination with a transformer-based classifier (BERTje) is feasible. The resulting model outperforms (62.95% F-score) a pipeline of a CRF approach to event trigger word detection and a BERT-based event classifier. We also demonstrated that a syntactic clause can be used as event span. Prominence classification is our answer to take into account a real-world situation where event types in held-out test data are frequently not covered because of training data scarcity. The BERTje model benefits from self-attention heads with linguistic notions such as syntax and coreference and outperformed (70.75% F-score) an SVM baseline classification model. A bottleneck of classifying prominence labels only based on the sentence level, is the lack of context. This has an impact mainly on the Background prominence class. Therefore further work includes exploring ways to provide more context information in the transformer model. It can be fine-tuned on training data where previous and following annotated events to the current single event instances are inserted. As a next step the BERTje self-attention mechanism will be leveraged to select the tokens in the syntactic clause with the highest attention values. This will allow e.g. the generation of nominal constituents on top of the clauses generated by the syntactic parser. Although the transformer model exhibits promising performances fine-tuned on a small dataset, data augmentation of the training set might optimize the fine-tuning and boost performances. Finally the classifier output will be fed into a news recommender system.

Acknowledgements

We thank the reviewers for their valuable comments. This work was supported by Ghent University under grant BOFGOA201800601.

References


Camiel Colruyt, Orphée De Clercq, and Véronique Hoste. under review. Eventdna: a dataset for dutch news event extraction as a basis for news diversification. Under review.


Teruko Mitamura, Zhengzhong Liu, and Eduard H Hovy. 2015a. Overview of taccbp 2015 event nugget track. In TAC.


Gertjan Van Noord et al. 2006. At last parsing is now operational. In TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles, pages 20–42.


Automatic Detection and Classification of Mental Illnesses from General Social Media Texts

Anca Dinu¹ and Andreea-Codrina Moldovan²

¹Faculty of Foreign Languages and Literatures, Digital Humanities Research Centre, University of Bucharest
²Faculty of Foreign Languages and Literatures, University of Bucharest
ancaddinu@gmail.com, moldovanandreeacodrina@gmail.com

Abstract

Mental health is getting more and more attention recently, depression being a very common illness nowadays, but also other disorders like anxiety, obsessive-compulsive disorders, feeding disorders, autism, or attention-deficit/hyperactivity disorders. The huge amount of data from social media and the recent advances of deep learning models provide valuable means to automatically detecting mental disorders from plain text. In this article, we experiment with state-of-the-art methods on the SMHD mental health conditions dataset from Reddit (Cohan et al., 2018). Our contribution is threefold: using a dataset consisting of more illnesses than most studies, focusing on general text rather than mental health support groups and classification by posts rather than individuals or groups. For the automatic classification of the diseases, we employ three deep learning models: BERT, RoBERTa and XLNET. We double the baseline established by Cohan et al. (2018), on just a sample of their dataset. We improve the results obtained by Jiang et al. (2020) on post-level classification. The accuracy obtained by the eating disorder classifier is the highest due to the pregnant presence of discussions related to calories, diets, recipes etc., whereas depression had the lowest F1 score, probably because depression is more difficult to identify in linguistic acts.

1 Introduction

An analysis performed by Chisholm et al. (2016) estimates that approximately 10% of the world’s population is living with a mental illness. The Global Burden of Disease Study (2017) states that depression is a very common illness and there are more than 264 million people affected by it. At its worst, the illness can lead to suicide and it is the second highest cause of death for people between 15 and 29. Between 76% and 85% of the potentially diagnosed people, do not benefit from any treatment for their illness due to living in impoverished areas and not having access to mental care. It is difficult to discuss about digital solutions in the context of isolated areas with low data availability and limited access to professional help. Social stigma is another obstacle present regardless of age, gender or race, which makes early intervention difficult. Persons facing difficulties often avoid discussing their issues from various reasons. However, researchers working with Machine Learning algorithms can draw plenty of expertise from the unstructured data roaming the World Wide Web. The advent of social media platforms brings up an influx of large quantities of various types of unstructured textual data. The continuous advancements made in the field of Machine Learning enable the possibility to analyse such volumes of data efficiently. Experiments in this interdisciplinary domain managed to bring up useful input for mental health practitioners, socio-linguists, computer scientist and other researchers in the field. Pennebaker et al. (2015) perform one of the most influential quantitative studies, which reveals the way patterns of parts of speech, as labelled by LIWC founders, correlate with types of personalities and types of mental illnesses. The classes and the psychological dimensions mapped together served as a start for many projects including the prediction of Dark Triad personality traits by Sumner et al. (2012) and the risk of self-harm by Soldaini et al. (2018). Research in the area is conducted mainly on texts from mental health support groups, on just a few illnesses and some groups of individuals.

Our main research questions for this article are if and to what extent it is possible to detect and classify mental illnesses from general texts, if there...
are any differences between the difficulty of automatic detection and classification of different illnesses and, finally, if such a classification may rely on posts only. To this end, we experiment with state-of-the-art methods on Reddit, to improve previous results and provide new insights on mental illness discovery from general text. Reddit is a social media network hosting numerous communities where users join in order to participate in various discussions. Each community or “board”, the way it is called by Reddit users, has a subject on which people must post. We employed highly performant deep learning models such as the Transformers, introduced by Vaswani et al. in 2017, which also led to the creation of BERT by Devlin et al. in 2018, a pretrained model trained on expansive general datasets, in order to be later fine-tuned on more specific tasks. Vale et al., (2021) efficiently applied BERT for question answering. Topal et al., (2021) use it for text generation and Sun et al., (2020) use it for text classification. Thus, we identify a solid ground for efficient usage in mental illnesses detection and classification.

2 Related Work

NLP researchers have shown an increased interest in the area at the intersection of Machine Learning and Psychiatry in the last years. Social media is an indispensable resource for research. Yet, the particularities of the online setting rise a range of challenges. As there are not any standards established for using social data, practitioners from many fields pointed to the dangers of using such data without a clear framework. Olteanu et al., (2019) address the issue of “biases, methodological pitfalls, and ethical boundaries” - discussing the problems often left unaddressed by researchers working with this kind of data. Selbst et al. (2019) analyse not only the ethical dilemma revolving around this type of studies, but also their feasibility and the integration of the social component into the compound of a socio-technical system.

When it comes to detecting mental illnesses from social media data, we have many examples at hand, which often look at data coming from those Reddit communities, which are support groups for people struggling with an illness or another. Most articles look at a single illness in comparison to a control group: Vedula et al. (2017) and Tsugawa (2019) – depression, and Bimbaum et al. (2020) – schizophrenia. Our goal is to detect a wide range of mental illnesses using deep learning techniques, which seem like the best candidates for this task. Jiang et al. (2020) employ deep learning methods similar to ours, but we concentrate on obtaining better results by training the models on individual posts rather than posts grouped by users, which might not work as expected. For example, if a user produced few contributions or has a fresh account, they would probably have few posts available. On the other hand, some types of user are the observing type and rarely contribute to discussions. One aspect worth mentioning is the nature of the data used in many classification tasks. Texts containing explicit content and linguistic cues pertaining to the properties of a certain illness are often used. Kim et al. (2020) and Thorstad et al. (2019) perform automatic text classification by their author’s mental illnesses, with good results, on texts that specifically discussed these conditions on dedicated forums. Nevertheless, these classifications are of little help in finding risk population, when looking at general text, which does not include mental illness topics. Among the few researchers who report using datasets containing general discussions coming from people who self-reported their diagnosis in one of the support communities are Jiang et al. (2020) and Cohan et al. (2018).

The results are favorable and leave room for improvement. We believe it is important to experiment further for a better understanding of the ways in which mental illnesses can be detected in earlier stages and how even general discussions contain traces of how mental illnesses manifest themselves in language. In addition, this is a direction worthy of exploration because the persons asking for guidance represent a very small and idiosyncratic part of the population battling with mental illnesses, thus early mental illness detection from general text might be of a real help.

3 Data

We used the SMHD dataset introduced by Cohan et al. (2018). This dataset contains non-explicit texts: A Large-Scale Resource for Exploring Online Language Usage for Multiple Mental Health Conditions. They test some classification algorithms, but no deep learning. Also, employed LIWC categories for classification. These categories include standard linguistic dimensions – pro-nouns, articles, present tense, future tense; psychological processes – positive emotions,
negative emotions, anger, anxiety; personal concerns – work, achievements. The SMHD dataset contains texts extracted from Reddit’s general discussion communities grouped on users and illnesses. Individuals diagnosed with a mental illness were detected by searching for self-reports in the dedicated support groups. The dataset features multiple illnesses, which are present in the psychiatric taxonomy DSM-5 (American Psychiatric Association, 2013). As stated by the authors of the dataset, “Six conditions are top-level DSM-5 disorders: schizophrenia spectrum disorders (schizophrenia), bipolar disorders (bipolar), depressive disorders (depression), anxiety disorders (anxiety), obsessive-compulsive disorders (ocd) and feeding and eating disorders (eating). The three other conditions are one rank lower: post-traumatic stress disorder (ptsd) is classified under trauma- and stress-related disorders, and autism spectrum disorders (autism) and attention-deficit/hyperactivity disorder (adhd) under neurodevelopmental disorders”. The opposing group of users is the control one, whose members are selected based on having no posts in the support groups and at least 50 posts on Reddit. The complete dataset contains 20,406 diagnosed users and 335,952 control users. The texts do not contain any terms related to mental health, neither the diagnosed groups, nor the control ones. Our experiments will use just a selection of each group of illnesses to speed up the computation process. The models are not user centered and will learn from each individual post. Selecting data based on a fixed number of users was not suitable for our tasks due to the imbalance at the user level when it comes to the number of comments and posts available. Therefore, we selected randomly 50,000 posts for each group of users.

The numbers shown in tables 1 and 2 might reflect certain particularities about an illness, how the diagnosed users communicate in the online environment. This variation depends also on how the users engage, whether they create posts or comment on somebody else’s and on the format adopted by each community – if pictures are posted often, then the comments are on the shorter side, if story telling is the center of the community, people engage with the purpose of telling their opinion or a similar story, hence the lengthier texts.

The authors of the dataset conducted a linguistic analysis based on LIWC categories. Several differences were observed between the diagnosed groups and the control users. Pennebaker et al. (2015) and Ireland and Mehl (2014) underline that pronounced usage of first-person singular with most conditions is consistent with the theory that illness drives one towards self-focus. An interesting finding underlining the bias of the dataset towards the predominantly male demographic is the female references that point to discussions about relationships and love related issues with the bipolar, depression and anxiety groups.

<table>
<thead>
<tr>
<th>Illness</th>
<th>No. of users</th>
<th>No. of comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depression</td>
<td>500</td>
<td>71017</td>
</tr>
<tr>
<td>ADHD</td>
<td>500</td>
<td>73201</td>
</tr>
</tbody>
</table>

Table 1: The number of comments produced by the two groups are similar.

<table>
<thead>
<tr>
<th>Illness</th>
<th>No. of users</th>
<th>No. of comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTSD</td>
<td>300</td>
<td>40885</td>
</tr>
<tr>
<td>Eating</td>
<td>300</td>
<td>10526</td>
</tr>
</tbody>
</table>

Table 2: The number of comments produced by the two groups of users is not balanced.

<table>
<thead>
<tr>
<th>Illness</th>
<th>Total No. of tokens per group</th>
<th>Mean no. of tokens per group</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPR</td>
<td>3,246,814</td>
<td>38.11</td>
</tr>
<tr>
<td>ANX</td>
<td>3,304,634</td>
<td>24.16</td>
</tr>
<tr>
<td>BIP</td>
<td>3,266,525</td>
<td>38.54</td>
</tr>
<tr>
<td>EAT</td>
<td>2,206,672</td>
<td>42.92</td>
</tr>
<tr>
<td>ADHD</td>
<td>3,241,564</td>
<td>40.47</td>
</tr>
<tr>
<td>PTSD</td>
<td>3,558,287</td>
<td>46.42</td>
</tr>
<tr>
<td>SCHIZO</td>
<td>4,611,530</td>
<td>37.10</td>
</tr>
<tr>
<td>OCD</td>
<td>3,068,948</td>
<td>42.01</td>
</tr>
<tr>
<td>AUT</td>
<td>3,348,654</td>
<td>39.24</td>
</tr>
</tbody>
</table>

Table 3: The number of tokens per group of illnesses and average number of tokens per person in a given group.

Reddit does not impose a very strict post limit hence we have diverse lengths. However, the deep learning models we used impose a limit for training. The SMHD has already undergone preprocessing, but we needed more cleaning. We remove any posts shorter than 4 tokens. Very short texts are often noise like thankful comments or very short approval phrases, which would confuse the model and do not contain significant meaning.
actions are pertaining to a class or another. Special characters and symbols are removed, and contractions are handled using a dictionary automatically translating them into the expanded forms. Table 3 shows the average number of tokens produced by users diagnosed with a certain illness and the average number of tokens a post from a user has. The next section will look at another data related problem, namely the ethics and biases of working with social media data.

4 Ethics and Biases

Reddit represents a social media application whose users are part of communities and engage in discussions. Each social media network represents a cluster of people who are defined by certain characteristics. The Hootsuite yearly report (2021) shows that more than 60% of the Reddit users are males aged 18 to 34. Accordingly, studies show that there is a tendency in males to display less emotionally charged input due to the social stigma in the offline world. Nadeau et al. (2016) find that men often avoid seeking professional help or talking about their problems. Concealing their emotional state in real life is a strategy in order to avoid prejudices and is not something specific for the female population. Ireland and Mehl’s (2014) research conducted in the psychology area proves that manifestations of negative emotions are muted across many settings and situations. Alternatively, Schoenebeck (2013) and Shelton et al. (2015) demonstrate that people tend to discuss personal things in anonymous spaces and share unpopular opinions. In this situation, Reddit represents a good source of data for a population, which is underrepresented in clinical studies. Anderson (2015) prove that some platforms might be more attractive for a demographic than others. Behavioral biases imply that users of a platform display a particular behavior, observable in how they interact with each other or what type of content they create. One such bias is the way in which users seek and share information. De Choudhury et al. (2014) discovered that users diagnosed with one illness behave differently in this aspect from the others. Nevertheless, we cannot claim that this is representative for all the individuals diagnosed with a mental illness. There are certain biases plaguing the studies based on social media, which should be at least mentioned for awareness. Here, we consider the population bias a positive fact, which enables studies targeting the young adult and adult males. However, this bias does not affect much our dataset, because the data collected comes from neutral communities where a variety of topics is discussed.

5 Discriminative Features

We run a Naïve Bayes Classifier in order to find out the most informative features from each category in our dataset. We used the classifier implemented in the scikit-learn library by Pedregosa, et al. (2011) to get a top of n most informative words by scores. Our experiment includes the 9 illnesses as labels, and the control group. The top n words can be seen in table 4. We notice that across the dataset, the top 8 words are mostly associated with the illness’s groups. The words belong to the category of meaning words as established by the LIWC taxonomy. Even if the texts come from general discussion boards with less restrictive discussion topics, we can see that the terms are related to the diagnosis. There are words pertaining to the group of emotion: “awkwardness”, “spiritual”, “psychotic”, “guilt”; love and sexuality: “attraction”, “sexuality”, “cis”, “trans”, “hormones”; terms related to illness and medication: “illness”, “doses”, “medication”, “therapy”, “pharmacist”, “relapse” and acronyms: “NC” (no contact), “AA” (Alcoholics Anonymous), HSV (Herpes Simplex Virus), STD (Sexually Transmitted Diseases), TRP (in this case it refers to a Reddit community called r/The Red Pill – a controversial Men’s Rights Activists (MRA) space which has now been removed). The idiosyncrasy of the network is seen in the occurrence of terms related to its own internal structure. Wide ranges of topics are discussed and that might differ in a clinician’s office, in face-to-face situations and where one’s identity is known. These distinctive features will help our classifiers to better distinguish the control users from the diagnosed ones because in some cases like in the depressed users and the ones suffering from anxiety the language is less distinctive.

6 Classification Methods

Identifying significant differences between our groups was the main drive for training classifiers. We trained 3 different models based on the Transformers architecture to see how each performs binary classification between a diagnosed group and a control one. We obtained state-of-the-
art results for text classification using BERT, RoBERTa and XLNET, as it follows.

BERT is a language representation model designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both the left and right context in all layers introduced by Devlin et al. (2018). BERT’s key technical innovation is applying the bidirectional training of Transformer, which is an advanced Long Short Term Memory Network (LSTM) to language modelling. The model uses the Transformer architecture to capture long distance dependencies within sentences. The pre-trained model contains a general knowledge of language and by giving it task-specific data we can obtain promising results. BERT uses a special tokenizer, which has a specific way of dealing with words outside its vocabulary. It accepts input in a required formatting: we have to add tokens marking the start and end of a sentence, pad and truncate sentences in order to have a uniform length. An attention mask is added in order to differentiate between the padding and the first type of tokens. The [CLS] token marks the beginning of a sentence and must be present for any sentence-level classification task. The [SEP] token separates one sentence from the next so the model can learn entailment. [CLS] and [SEP] tokens were used when the developers pretrained BERT, and we must preserve the same scheme for the model to work properly. The maximum sentence length supported by BERT is 512 tokens. Our experiment uses the Hugging Face Pytorch implementation.

<table>
<thead>
<tr>
<th>Illness</th>
<th>Chi ill</th>
<th>Chi Cont</th>
<th>ADHD</th>
<th>Chi ill</th>
<th>Chi Cont</th>
<th>Eating</th>
<th>Chi ill</th>
<th>Chi Cont</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTSD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NC</td>
<td>80.9</td>
<td>1.0</td>
<td>kratom</td>
<td>56.4</td>
<td>1.0</td>
<td>carbs</td>
<td>71.0</td>
<td>1.0</td>
</tr>
<tr>
<td>TRP</td>
<td>58.1</td>
<td>1.0</td>
<td>NC</td>
<td>41.3</td>
<td>1.0</td>
<td>deficit</td>
<td>53.0</td>
<td>1.0</td>
</tr>
<tr>
<td>abusive</td>
<td>53.8</td>
<td>1.0</td>
<td>backyard</td>
<td>22.6</td>
<td>1.0</td>
<td>calorie</td>
<td>47.5</td>
<td>1.0</td>
</tr>
<tr>
<td>meetings</td>
<td>51.5</td>
<td>1.0</td>
<td>attraction</td>
<td>21.8</td>
<td>1.0</td>
<td>pound</td>
<td>42.9</td>
<td>1.0</td>
</tr>
<tr>
<td>therapy</td>
<td>36.0</td>
<td>1.0</td>
<td>pharmacist</td>
<td>20.9</td>
<td>1.0</td>
<td>obese</td>
<td>42.0</td>
<td>1.0</td>
</tr>
<tr>
<td>pregnancy</td>
<td>26.1</td>
<td>1.0</td>
<td>allergy</td>
<td>20.0</td>
<td>1.0</td>
<td>oily</td>
<td>42.0</td>
<td>1.0</td>
</tr>
<tr>
<td>boundaries</td>
<td>35.2</td>
<td>1.0</td>
<td>childfree</td>
<td>26.0</td>
<td>1.0</td>
<td>guilt</td>
<td>28.2</td>
<td>1.0</td>
</tr>
<tr>
<td>hobbies</td>
<td>31.4</td>
<td>1.0</td>
<td>hormonal</td>
<td>20.3</td>
<td>1.0</td>
<td>rice</td>
<td>24.4</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Illness</th>
<th>Chi ill</th>
<th>Chi Cont</th>
<th>Anx</th>
<th>Chi ill</th>
<th>Chi Cont</th>
<th>Bip</th>
<th>Chi ill</th>
<th>Chi Cont</th>
</tr>
</thead>
<tbody>
<tr>
<td>caffeine</td>
<td>93.3</td>
<td>1.0</td>
<td>HSV</td>
<td>72.4</td>
<td>1.0</td>
<td>Meds</td>
<td>59.2</td>
<td>1.0</td>
</tr>
<tr>
<td>meds</td>
<td>38.0</td>
<td>1.0</td>
<td>bernie</td>
<td>61.1</td>
<td>1.0</td>
<td>IGN</td>
<td>1.0</td>
<td>58.4</td>
</tr>
<tr>
<td>attraction</td>
<td>34.1</td>
<td>1.0</td>
<td>Eyeshadow</td>
<td>40.2</td>
<td>1.0</td>
<td>Kratom</td>
<td>45.7</td>
<td>1.0</td>
</tr>
<tr>
<td>flirting</td>
<td>33.4</td>
<td>1.0</td>
<td>Cannabis</td>
<td>37.4</td>
<td>1.0</td>
<td>Sweetheart</td>
<td>41.8</td>
<td>1.0</td>
</tr>
<tr>
<td>hormones</td>
<td>31.1</td>
<td>1.0</td>
<td>boyfriends</td>
<td>27.0</td>
<td>1.0</td>
<td>Bondage</td>
<td>38.0</td>
<td>1.0</td>
</tr>
<tr>
<td>symptoms</td>
<td>26.1</td>
<td>1.0</td>
<td>kratom</td>
<td>33.6</td>
<td>1.0</td>
<td>Vaping</td>
<td>29.3</td>
<td>1.0</td>
</tr>
<tr>
<td>Mii</td>
<td>1.0</td>
<td>27.0</td>
<td>STD</td>
<td>26.0</td>
<td>1.0</td>
<td>Literature</td>
<td>27.4</td>
<td>1.0</td>
</tr>
<tr>
<td>cigars</td>
<td>1.0</td>
<td>29.3</td>
<td>sexuality</td>
<td>25.3</td>
<td>1.0</td>
<td>hormones</td>
<td>23.6</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Illness</th>
<th>Chi ill</th>
<th>Chi Cont</th>
<th>OCD</th>
<th>Chi ill</th>
<th>Chi Cont</th>
<th>Autism</th>
<th>Chi ill</th>
<th>Chi Cont</th>
</tr>
</thead>
<tbody>
<tr>
<td>spiritual</td>
<td>84.6</td>
<td>1.0</td>
<td>trans</td>
<td>133.4</td>
<td>1.0</td>
<td>feminine</td>
<td>75.0</td>
<td>1.0</td>
</tr>
<tr>
<td>psychotic</td>
<td>76.1</td>
<td>1.0</td>
<td>therapy</td>
<td>81.7</td>
<td>1.0</td>
<td>sensory</td>
<td>64.5</td>
<td>1.0</td>
</tr>
<tr>
<td>meditation</td>
<td>49.9</td>
<td>1.0</td>
<td>feminine</td>
<td>51.3</td>
<td>1.0</td>
<td>trans</td>
<td>57.5</td>
<td>1.0</td>
</tr>
<tr>
<td>consciousness</td>
<td>44.8</td>
<td>1.0</td>
<td>NC</td>
<td>44.5</td>
<td>1.0</td>
<td>NC</td>
<td>50.9</td>
<td>1.0</td>
</tr>
<tr>
<td>cannabis</td>
<td>39.1</td>
<td>1.0</td>
<td>cis</td>
<td>31.4</td>
<td>1.0</td>
<td>relapse</td>
<td>29.9</td>
<td>1.0</td>
</tr>
<tr>
<td>doses</td>
<td>37.0</td>
<td>1.0</td>
<td>TRP</td>
<td>29.4</td>
<td>1.0</td>
<td>TRP</td>
<td>27.8</td>
<td>1.0</td>
</tr>
<tr>
<td>literature</td>
<td>37.7</td>
<td>1.0</td>
<td>scarring</td>
<td>29.4</td>
<td>1.0</td>
<td>awkwardness</td>
<td>24.6</td>
<td>1.0</td>
</tr>
<tr>
<td>AA</td>
<td>37.1</td>
<td>1.0</td>
<td>literature</td>
<td>28.1</td>
<td>1.0</td>
<td>illness</td>
<td>23.6</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 4: Chi-square scores per each illness and control group.
BertForSequenceClassification by Wolf et al. (2019). In order to setup this model, we experimented using different hyperparameters, loss functions, batch sizes and number of epochs. The authors of BERT recommend using it with the following specifications:

- batch sizes: 8, 16, 32, 64, 128;
- learning rates: 3e-4, 1e-4, 5e-5, 3e-5.

<table>
<thead>
<tr>
<th>Hyperparameters</th>
<th>BERT</th>
<th>RoBERTa</th>
<th>XLNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Length</td>
<td>256</td>
<td>256</td>
<td>126</td>
</tr>
<tr>
<td>Batch Size</td>
<td>3</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Weight Decay</td>
<td>0.0</td>
<td>0.0-0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>AdamWε</td>
<td>1e-5</td>
<td>1e-5</td>
<td>2e-5</td>
</tr>
<tr>
<td>AdamW β</td>
<td>0.9, 0.999</td>
<td>0.9, 0.999</td>
<td>0.9, 0.999</td>
</tr>
<tr>
<td>No. of training posts</td>
<td>100,000</td>
<td>100,000</td>
<td>100,000</td>
</tr>
<tr>
<td>No. of testing posts</td>
<td>20,000</td>
<td>20,000</td>
<td>20,000</td>
</tr>
</tbody>
</table>

Table 5: Hyperparameters for each model.

Our machine needed smaller batch sizes to be able to train the model, so we used 3. We established a learning rate (Adam ε) of 1e-5 for the AdamW loss function implemented by Loschilov and Hutter (2017). We trained the model for 3 epochs only, because we noticed overfitting starting with the 4th epoch.

The second method we used is XLNet, which is another method for pre-training language representations introduced by Yang et al. (2019). XLNet was meant to overcome the limitations imposed by BERT with its autoregressive model and does so by outperforming it on 20 tasks as shown by Yang et al. (2019). For this method, we have a different formatted input and there is no limit for the length of the input texts. However, the input arrays need to be of the same size. This is addressed by padding the inputs that do not meet the size of the longest sequence. Padding means simply adding 0s until the length is met. For this classifier we had to limit the length of sequences to 126 due to computational resources. The optimum batch size was 8. The loss function we used was AdamW with the same hyperparameters as for BERT. We trained this model for 4 epochs. With a training set of approximately 100000 texts, we get a number of 50000 training steps.

The last model we used, RoBERTa implemented by Liu et al. (2019), is Facebook AI’s training method and it promises to improve on BERT. The researchers involved in implementing RoBERTa prove that BERT was undertrained and there is still a long way to go in terms of design choices and the way in which the improvements are reported. We did not use the full size of our dataset due to its large size and subsequent long training times.

Finetuning RoBERTa implies loading the weights of the pretrained model, in our case, the RobertaForSequenceClassification model. We use a sequence length of 256 and a batch size of 8. The loss function used here is AdamW with Adam ε of 2e-5.

7 Results

We obtained the results using 50,000 posts for each group alike. The compound of 100,000 posts for each binary classifier was split in 80,000 for training and 20,000 for testing. We trained our models with different hyperparameters until we reached the optimum ones detailed in table 6. We manage to overrun the baseline established by Cohan et al. (2018) using Transformers-based models on just a sample of their dataset. Their best results lie at approximately 50% accuracy with 57% being the best result obtained using Supervised FastText on Bipolar Disorder, while ours lie at approximately 75%, with 81% the best result. We also improve the results obtained by Jiang et al. (2020) on post-level classification as seen in table 7 by a considerable margin. We compare the results obtained using BERT and calculate the difference between our results and the ones from Jiang et al. Higher results were obtained with XLNet and RoBERTa in some cases. The BERT model achieves the highest accuracy for an illness: schizophrenia, OCD, eating disorder, autism and anxiety. We notice that discriminative features play an important role in building a performant model. The accuracy obtained by the eating disorder classifier is the highest due to the pregnant presence of discussions related to calories, diets, recipes etc. (as seen in table 6), whereas for depression we obtained the lowest F1 score, probably because depression is not always identifiable in linguistic acts, cf. Ireland and Mehl (2014). It is often a matter of contextual factors that might drive a user to discuss their emotional state or any other.
Table 6: Results for BERT (B), XLNET (XL) and RoBERTa (R) classifiers. Best results are in bold for each illness.

| Metric | DEPR | CONT | SCHIZ | CONTR | OCD | CONT | EAT | CONT | BPD | CONT | ADHD | CONT | PTSD | CONT | AUT | CONT | ANX | CONT |
|--------|------|------|-------|-------|-----|------|-----|------|-----|------|------|------|------|------|-----|-----|-----|-----|-----|
| B      | 0.73 | 0.63 | 0.73  | 0.72  | 0.74 | 0.76 | 0.82 | 0.80 | 0.75 | 0.72 | 0.73  | 0.66 | 0.75 | 0.76 | 0.72 | 0.69 | 0.73 | 0.72 |
| R      | 0.70 | 0.66 | 0.71  | 0.74  | 0.72 | 0.77 | 0.83 | 0.78 | 0.74 | 0.73 | 0.71  | 0.68 | 0.76 | 0.75 | 0.71 | 0.70 | 0.73 | 0.72 |
| F1     | 0.68 | 0.73 | 0.75  | 0.81  | 0.73 | 0.70 | 0.75 | 0.77 | 0.70 | 0.73 | 0.70  | 0.69 | 0.76 | 0.70 | 0.72 | 0.71 | 0.71 | 0.73 |
| XL     | 0.74 | 0.57 | 0.68  | 0.69  | 0.73 | 0.70 | 0.73 | 0.87 | 0.76 | 0.69 | 0.71  | 0.67 | 0.76 | 0.76 | 0.70 | 0.69 | 0.74 | 0.71 |
| R      | 0.67 | 0.81 | 0.57  | 0.78  | 0.71 | 0.72 | 0.87 | 0.70 | 0.69 | 0.76 | 0.69  | 0.70 | 0.74 | 0.77 | 0.72 | 0.67 | 0.71 | 0.74 |
| F1     | 0.70 | 0.68 | 0.72  | 0.79  | 0.72 | 0.72 | 0.69 | 0.76 | 0.70 | 0.73 | 0.70  | 0.69 | 0.70 | 0.73 | 0.69 | 0.75 | 0.69 | 0.69 |
| R      | 0.71 | 0.59 | 0.73  | 0.66  | 0.77 | 0.65 | 0.78 | 0.74 | 0.73 | 0.70 | 0.68  | 0.76 | 0.75 | 0.74 | 0.70 | 0.70 | 0.77 | 0.73 |
| F1     | 0.68 | 0.70 | 0.72  | 0.78  | 0.75 | 0.71 | 0.75 | 0.70 | 0.75 | 0.70 | 0.70  | 0.73 | 0.72 | 0.71 | 0.71 | 0.73 | 0.73 | 0.73 |

8 Conclusions

Our automatic classification experiments used the Transformers-based models BERT, XLNet and RoBERTa on the SMHD dataset for the classification of 9 mental health illnesses. We manage to overrun Jiang et al. (2020) by approximately 0.10-0.15 on the single-post classification task and prove that individual posts yield satisfactory accuracy. We overrun Cohan et al. (2017) by 0.20-0.30 who did not employ any deep learning methods.

We used a Naïve Bayes Classifier to discover the most important features for each group of users. Our results add to the group of articles showing good prospects for this field. An encouraging finding is the sufficiency of focusing on general text rather than mental health support groups and classification by posts rather than individuals or groups. Another takeaway is the sufficiency of post-level classification and avenue to improve this approach in future work by paying attention to contextual cues such as time, events, entailment of posts or any other possible triggers that might help the earlier detection of a mental illness. Further experimentation with different setups and data that are more diverse is also required. This would benefit our research and increase the possibility of future integration of automated tools, which could assist clinicians in the earlier detection of mental health issues.

Table 7: Comparison of BERT classification results – Jiang et al. (2020), Cohan et al. (2017) and our model. We report the results obtained by our binary classifiers in comparison with the binary classifiers trained by them. Cohan et al. did not employ any deep learning methods at the time, so we picked the highest F1 scores obtained with SVMs, Logistic Regression, FastText and CNNs.

Acknowledgements

This research is supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI
References


Sumner, C., Byers, A., Boochever, R., Park G.J. “Predicting Dark Triad Personality Traits from Twitter Usage and a Linguistic Analysis of Tweets”, ResearchGate, doi:10.1109/IICMLA.2012.218.


A Pre-trained Transformer and CNN model with Joint Language ID and Part-of-Speech Tagging for Code-Mixed Social-Media Text

Suman Dowlagar  
LTRC  
IIIT-Hyderabad  
suman.dowlagar@research.iiit.ac.in

Radhika Mamidi  
LTRC  
IIIT-Hyderabad  
radhika.mamidi@iiit.ac.in

Abstract

Code-mixing (CM) is a frequently observed phenomenon that uses multiple languages in an utterance or sentence. There are no strict grammatical constraints observed in code-mixing, and it consists of non-standard variations of spelling. The linguistic complexity resulting from the above factors made the computational analysis of the code-mixed language a challenging task. Language identification (LI) and part of speech (POS) tagging are the fundamental steps that help analyze the structure of the code-mixed text. Often, the LI and POS tagging tasks are interdependent in the code-mixing scenario. We project the problem of dealing with multilingualism and grammatical structure while analyzing the code-mixed sentence as a joint learning task. In this paper, we jointly train and optimize language detection and part of speech tagging models in the code-mixed scenario. We used a Transformer with convolutional neural network architecture. We train a joint learning method by combining POS tagging and LI models on code-mixed social media text obtained from the ICON shared task.

1 Introduction

In bilingual and multilingual communities, code-mixing or code-switching occurs when a person alternates languages below the phrase level inside a sentence or an utterance. Code-mixing (CM) is generally observed in informal settings such as casual conversations or social media text.

Code-mixing is defined as mixing phrases, words, and morphemes of one language into another language (Myers-Scotton, 1997).

Language identification (LI) and part of speech (POS) tagging are the fundamental steps in processing any code-mixed sentence. LI deals with resolving the language ambiguity of each word in a code-mixed text. POS tagging involves assigning a part of the speech label for each word in a sentence based on its syntactic and semantic information. It helps analyze the grammatical structure of the sentence. Both Language Identification and POS tagging are sequence labeling tasks as they tag each word in a sentence by its corresponding language and POS tags, respectively.

While processing a monolingual text, the primary step to understand the sentence’s grammatical structure would be POS tagging. However, in the code-mixed scenario, we must consider multilingual phenomena, i.e., each word’s language while POS tagging. Similarly, POS tagging helps capture a better grammatical structure of the text. Moreover, identifying the grammatical structure can improve language identification. In the code-mixing scenario, two tasks go hand in hand. Thus, a joint learning model on POS tagging and LI will considerably enhance the code-mixed text analysis.

Recently the transformer models with transfer learning such as BERT (Devlin et al., 2018) achieved state-of-the-art accuracy in sequence classification tasks. An evaluation benchmark on code-mixed datasets - GLUECoS (Khanuja et al., 2020) stated that a modified version of BERT (Devlin et al., 2018) called mod-mBERT that was fine-tuned on synthetically generated code-switched data outperformed on all the code-mixed datasets.

Recently, BERT with ensemble models has shown improved performance on text classification tasks (Dowlagar and Mamidi, 2021; Safaya et al., 2020). To improve our tasks’ performance, we have used a convolutional model. The convolutional approach learns a compositional structure in the sequences more efficiently since the representations are built on hierarchy.

In the code-mixed social media text, the native words are often written in the Roman script. The introduction of such non-standard transliterations will add complications while processing the
text. It is necessary to deal with these complications as they might introduce errors in further processing steps. To tackle such problems, we back-transliterated the Roman words into their native script.

This paper presents a pre-trained transformer encoder with convolutional neural network architecture for POS tagging and language identification of Code-Mixed Social-Media text. The model uses sub-word level input representation to handle morphologically rich words.

Our contributions are as follows:

1. We pre-process the data to deal with variations in spelling and transliterations.
2. We propose a transfer learning-based approach to jointly model LI and POS tagging tasks, achieving state-of-the-art accuracy on the ICON 2016 shared task dataset.
3. We design a BERT with convolutional neural network architecture for LI and POS tagging tasks. Our analyses confirm it is a better alternative than the joint Bi-LSTM model.

The paper is organized as follows. Section 2 presents a survey of related works on language identification and POS tagging in a code-mixed social media text. Section 3 introduces the proposed approach of jointly modeling the language identification and the part-of-speech tagging task for code-mixed social media text. In Section 4, we present the experimental setup and performance of our joint model. Section 5 concludes the work.

2 Related Work

Since the last decade, LI and POS tagging for code-mixed text has been a topic of interest in the field of natural language processing (NLP).

CM language identification: (Aguilar and Solorio, 2019) used a large pre-trained model ELMo, and adapted it to code-switching settings to obtain contextually rich embeddings. The paper used a Bi-LSTM CRF for language identification. (Mave et al., 2018) compared different word-level language identification systems for code-switched Hindi-English data and a standard Spanish-English dataset and found that the CRF model works the best on the given datasets. (Gundapu and Mamidi, 2018) performed LI using CRF on Telugu-English CM data. (Barman et al., 2014; Solorio et al., 2014) used SVM and CRF for language identification on CM data. (Rijhwani et al., 2017) used HMM for language identification on seven languages.

CM POS tagging: (Bhattu et al., 2020a) addressed the problem of prediction of POS tags for OOV words in low resource languages using character-based word embedding as input features to a Bi-LSTM and CRF network. (Ball and Garrette, 2018) used a meta embedding approach for the part of speech tagging where the word is represented in both code-mixed languages. Thus, it maintains embeddings for each language and is processed appropriately at inference time without committing to one or the other language. It used a Bi-LSTM model for POS tagging. (Jamatia et al., 2015) presented POS tagging results on English-Hindi social media text using various ML and deterministic approaches, among which CRF gave the best results.

Joint Learning: The joint learning models have been studied for the code-switched LI and POS scenarios. (Soto and Hirschberg, 2018) proposed a joint learning approach for POS tagging and LI using recurrent neural networks. (Barman et al., 2016) used a factorial CRF for joint modeling of POS tagging and language identification.

To the best of our knowledge, a joint learning model with BERT and CNN architecture for language identification and POS tagging on code-mixed data is not yet analyzed.

3 Proposed Model

In this section, we briefly describe the BERT and convolutional models. We then introduce the proposed joint model for POS tagging and LI of CM social media text. The architecture of the proposed model is given in figure 1.
3.1 Pre-processing

This section presents the steps performed for transliterating the code switched data to its respective languages.

The given code-mixed dataset portrays the real-time scenario of variations in script changes. Pre-processing is necessary on the text. During pre-processing,

1. To resolve the transliteration variations resulting from script change, we back-transliterated the script to the native language. Our code-mixed datasets have the matrix and embedded languages, where the embedded language is mostly English. Firstly, we used the NLTK\(^1\) English word corpus to detect if the word is in English or not. We used google trans API\(^2\) to detect the word’s language id. Later, we back transliterated the non-English word to its native script using a deep transliteration engine\(^3\).

3.2 Background - BERT

Bi-directional Encoder Representations with Transformers (BERT) (Devlin et al., 2018) is a transformer encoder stack trained on the large corpora. BERT uses a transformer architecture (Vaswani et al., 2017). The transformer architecture consists of a series of multi-headed attention, point-wise feed-forward layers with layer normalization to learn contextual relations between words (or sub-words) in a text. In our approach, we used a small version of the pre-trained multilingual BERT model called bert-base-multilingual-cased obtained from the transformers library. The pre-trained multilingual BERT models are trained on a large multilingual Wikipedia and book corpus. They capture a better semantic representation of words in a text. As the pre-trained model is trained on generic corpora, we need to fine-tune the model for our tasks. During fine-tuning, the pre-trained BERT model parameters are updated.

3.3 Convolutional Model

The convolutional neural network (CNN) model is made of convolutional layers. In short, a convolutional layer uses filters. These filters have a width for processing text. If a filter has a width of 3, then it can see three consecutive tokens. It has many filters, and these filters will slide across the sequence, from beginning to the end, looking at all three consecutive tokens at a time. These filters will learn to extract a different feature from the text. This feature extraction will then be used by the model - potentially as input to another convolutional layer.

Similar to the BERT model, the convolutional model has a positional embedding layer to remember the sequence. The token and positional embeddings are element-wise summed together to get a vector containing information about the token and its position within the sequence. It is followed by a linear layer that transforms the embedding vector into a vector with the required hidden dimension size. The next step is to pass this hidden vector into convolutional blocks. In convolutional blocks, the input sequence is padding such that the length of the input sequence and output sequence should be equal. A Special activation function called gated linear units (GLU) (Dauphin et al., 2017) is used after convolutions. The GLUs have gating mechanisms (similar to Bi-LSTMs (Hochreiter and Schmidhuber, 1997)) contained within the activation function. After passing through the GLU activation, each token’s hidden dimension size is the same as it was when it entered the convolutional blocks. Finally, residual connections are applied to solve the vanishing gradients problem, and linear transformations are done to match the dimensions. This process is repeated for $N$ convolutional blocks.

CNN model is formulated as,

$$ h^l_i = v^l \left( \sum_{k=0}^{l-1} h^{l-1}_{i-k/2}, \ldots, h^{l-1}_{i+k/2} \right) + b^l_w + h^{l-1}_i $$

(1)

Where $h^l_i$ is the output of the $i^{th}$ sequence in $l^{th}$ block. $v^l$ is the GLU activation function. $\left[ h^{l-1}_{i-k/2}, \ldots, h^{l-1}_{i+k/2} \right]$ are convolutional transformations of previous layer. $W^l$ and $b^l_w$ are learnable parameters and $h^{l-1}_i$ is the residual output from the previous layer.

After performing the convolutions, Finally, we compute a distribution over the $T$ possible LI or POS tags by transforming the top convolutional encoder output $h^L$ via a linear layer with weights $W_{s_2}$ and bias $b_{s_2}$.

$$ y^l_{i} = \text{softmax} \left( W_{s_2} h^L_i + b_{s_2} \right) $$

(2)

$$ O^l_{i} = \text{argmax} \left( y^l_{i} \right) $$

(3)
3.4 Joint Learning Framework

In sequence tagging, we have to learn a function \( f : x \rightarrow y \) that maps an input sequence \( x \) to the corresponding label sequence \( y \). We want to find the best label sequence \( y \) given an input sequence \( x \) that maximizes the probability (\( p \)) of the sequence given the label.

\[
\hat{y} = \arg\max_y p(y|x)
\]

In the multilingual code-mixed scenario, if the POS tagging and LI tasks are trained together, joint learning will help the model learn better data representation, improving the tagging accuracy. Thus, we propose a joint training model. The LI and POS tagging tasks can be viewed as conditionally independent in the joint learning scenario, given the transformer encoder parameter values. To jointly train the POS tagging and LI models, the objective is formulated as,

\[
p(y_{\text{lang}}, y_{\text{pos}}|x) = \prod_{t=1}^{T} p(y_{\text{lang}}|x,E)p(y_{\text{pos}}|x,E)
\]

Where \( p(y_{\text{lang}}, y_{\text{pos}}|x) \) is the conditional probability of learning the joint task given the input word sequence, and \( E \) stands for encoder parameter values.

The negative logarithm of the above equation gives us the loss function:

\[
L = -\log \prod_{t=1}^{T} p(y_{\text{lang}}|x,E)p(y_{\text{pos}}|x,E)
\]

\[
= -\log \prod_{t=1}^{T} p(y_{\text{lang}}|x,E) - \log \prod_{t=1}^{T} p(y_{\text{pos}}|x,E)
\]

Which can be further written as

\[
L = L_{\text{lang}} + L_{\text{pos}} \tag{4}
\]

If we apply a joint loss function for both the models, we can learn a better POS tagging and LI model for the CM scenario.

4 Experiments

This section evaluates the above method with the individual POS tagging and language identification models on the CM dataset.

<table>
<thead>
<tr>
<th>Comment</th>
<th>Language id</th>
<th>POS tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>@buttmona098</td>
<td>univ</td>
<td>@</td>
</tr>
<tr>
<td>@Hamzaldrees</td>
<td>univ</td>
<td>@</td>
</tr>
<tr>
<td>accha</td>
<td>hi</td>
<td>N_NNP</td>
</tr>
<tr>
<td>topic</td>
<td>en</td>
<td>JJ</td>
</tr>
<tr>
<td>change</td>
<td>en</td>
<td>N_NN</td>
</tr>
<tr>
<td>karo</td>
<td>hi</td>
<td>V_VM</td>
</tr>
</tbody>
</table>

Table 1: A example of language and POS tagged Hindi-English code-mixed sentence obtained from the corpus. Where "hi" refers to the Hindi language and "en" reference to the English language.

4.1 Dataset

The dataset used for the LI and POS tagging task on the CM language is obtained from the ICON shared task 4. The CM social media text consists of 3 languages (Bengali, Hindi, and Telugu with mixed English words). The sentences are in roman script. Each word is labeled with its corresponding language label and POS tag. The total number of CM sentences in this dataset is 9212. Two types of POS tagging schemas are used in the dataset. One is a coarse-grained (CR) tagset that used google’s universal POS tag-set (Petrov et al., 2011) and the other is fine-grained tagset (FI) with an extended tag-list related to social media text (Gimpel et al., 2010; Owoputi et al., 2013). We have used both POS tagsets in our experiments. The initial dataset contained errors in tagsets, which were addressed in the article (Bhattu et al., 2020b). We have used the revised tagset in our paper obtained from (Bhattu et al., 2020b). An tagged Hindi-English sentence is given in the table 1.

4.2 Baselines

We compared the performance of our model with the related works on the CM data. The acronym “(I)” refers to the individual model, and “(J)” refers to the joint model.

**CRF (I):** A CRF model is used for language identification on the CM data. The features set defined in (Gundapu and Mamidi, 2018) is used while training the CRF model.

**FCRF (J):** A factorial CRF model with joint learning (Barman et al., 2016) for LI and POS tasks.

**Bi-LSTM CRF (I):** This is the most commonly used model for the LI and POS tasks. It uses Bi-

LSTM and CRF consecutively for POS tagging the CM text (Aguilar and Solorio, 2019; Bhattu et al., 2020a).

**Bi-LSTM CRF (J):** A Bi-LSTM CRF model that is trained for the task of joint learning (Soto and Hirschberg, 2018).

**BERT (I):** Pre-trained multilingual BERT (Devlin et al., 2018) is used for the given LI and POS tasks.

**BERT (J):** A joint BERT model (Chen et al., 2019) is used for CM language identification and POS tagging.

### 4.3 Implementation

We implement the proposed model as follows. The input text is pre-processed. The pre-processed text is given to the BERT model. The BERT encodes the input text. The encoded input is given to the convolutional model. The encoded output is given to the 2 multi-layer perceptron models (MLP), each one for the two tasks LI and POS tagging. Then we aggregate the loss of the two models. The loss is then propagated backward, and the model optimizes to minimize the losses of both the NN models.

The proposed model is trained using Adam optimizer with cross-entropy loss obtained from the joint model. The hyperparameters are: Optimization ($\alpha$) = 0.001, dropout probability is 0.25. The number of epochs used is 10. All the deep learning models are implemented in python 3.6 using the PyTorch and the Torchtext libraries. We used the NVIDIA RTX 2070 graphics card with an 8GB GPU memory. We have used python-crfsuite\(^5\) and pytorch-crf\(^6\) for the CRF model, and the BERT model is obtained from the transformers\(^7\) library.

### 4.4 Performance

We compared our model with all the baselines and the results are tabulated in tables 2, 3 and 4. The proposed model has shown an improvement in the language identification and the POS tagging tasks compared to the baseline models (CRF and Bi-LSTM CRF). Using joint learning, The joint model has seen a further improvement when compared to the individual models.

Even the BERT model has proved better than the other models because of its state-of-the-art transformer architecture. We have observed that the transformer architecture helped the model to learn the tags that have a low frequency and gave better recall for the low-frequency tags. We observed the same in our convolutional seq2seq model.

Our model, the BERT with the convolutional seq2seq architecture, formed a meta-learning approach on the code-mixed text. The BERT model learned the better representation of the data. The kernels used in the convolutional seq2seq model helped the model consider the previous tagged information while predicting the current tag. We had observed that the suffixes are classified correctly when our approach was used.

For Language Identification, we have seen a considerable difference in joint learning and POS tagging when compared to the CRF model on the Telugu-English dataset. We have observed that most of the misclassified te(Telugu-word) and en(English-word) are correctly classified with joint learning. Even the acro(acronym) words which were of low frequency, were better identified with our approach. Some mixed words, i.e., with intra level code-mixing, are present in the data. These have undergone sub-word tokenization and were unidentifiable by the BERT model. As the CNN model uses the filter of size 3, the previous two tokens will be considered while predicting the tag of the current token. The CNN model and the pre-processing step helped the model to detect the mixed words correctly, which further improved the performance of our model.

In fine-grained and coarse-grained POS tagging, the major shift in joint-learning and individual task learning is observed in the baseline CRF model on the Bengali-English coarse-grained dataset. We have observed that the incorrect tagging of $G_N$ as $G_V$ or vice versa has decreased when joint learning is used for the POS tagging scenario.

We have also observed that our approach performed better in tagging $G_N$ and $G_V$ words when compared to the CRF and Bi-LSTM CRF models. It resulted in improved accuracy. It is due to its state-of-the-art multi-headed attention feature used in this model.

In the code-mixing scenario, the word’s POS tag depends on the following factors: how the word is used in the CM sentence, its multilingual feature, and its context. With the help of joint learning, which considered the grammatical structure and multilingual nature of the word, the incorrect classification problem of tagging $G_N$ as $G_V$ and
vice-versa is reduced.

We have observed low macro-F1 scores in our tagging models. It is due to the presence of distinct tags with low frequency and was insufficient to be trained by the existing models. These tags compromised the F1 score on the given data.

The experiments conducted on various models show that the joint learning model achieves improved POS tagging and LI in the code mixed scenario.

5 Conclusion

In this paper, we presented the joint BERT with the CNN model for POS tagging and LI. The joint learning model allowed the POS tagging and LI to be conditioned on each other to achieve better processing of code-mixed text. We tested our model with individual tasks. The results prove that our model achieves better metrics when compared to individual models. Such relations can effectively achieve better sentence-level semantic representa-

---

**Table 2:** Macro-F1 and Accuracy metric for Language Identification on CM social media data (Be stands for Bengali, En for English, Hi for Hindi, Te for Telugu)

<table>
<thead>
<tr>
<th>LI</th>
<th>Be-En</th>
<th>Hi-En</th>
<th>Te-En</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>macro-F1</td>
<td>Acc</td>
<td>macro-F1</td>
</tr>
<tr>
<td>CRF (I)</td>
<td>55.16</td>
<td>74.15</td>
<td>63.14</td>
</tr>
<tr>
<td>CRF (J)</td>
<td>55.29</td>
<td>74.80</td>
<td>63.46</td>
</tr>
<tr>
<td>Bi-LSTM + CRF (I)</td>
<td>56.49</td>
<td>81.87</td>
<td>63.40</td>
</tr>
<tr>
<td>Bi-LSTM + CRF (J)</td>
<td>56.80</td>
<td>82.47</td>
<td>63.56</td>
</tr>
<tr>
<td>mBERT (I)</td>
<td>58.17</td>
<td>89.81</td>
<td>63.31</td>
</tr>
<tr>
<td>mBERT (J)</td>
<td>58.20</td>
<td>90.20</td>
<td>63.51</td>
</tr>
<tr>
<td>pre-process + mBERT + CNN (I)</td>
<td>58.21</td>
<td>91.18</td>
<td>63.63</td>
</tr>
<tr>
<td>pre-process + mBERT + CNN (J)</td>
<td>58.25</td>
<td>91.21</td>
<td>63.71</td>
</tr>
</tbody>
</table>

**Table 3:** Macro-F1 and Accuracy metric for coarse-grained (CR) POS tagging on CM social media data (Be stands for Bengali, En for English, Hi for Hindi, Te for Telugu)

<table>
<thead>
<tr>
<th>POS (CR)</th>
<th>Be-En</th>
<th>Hi-En</th>
<th>Te-En</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>macro-F1</td>
<td>Acc</td>
<td>macro-F1</td>
</tr>
<tr>
<td>CRF (I)</td>
<td>52.76</td>
<td>65.93</td>
<td>56.38</td>
</tr>
<tr>
<td>CRF (J)</td>
<td>52.85</td>
<td>69.69</td>
<td>56.57</td>
</tr>
<tr>
<td>Bi-LSTM + CRF (I)</td>
<td>53.37</td>
<td>78.56</td>
<td>57.23</td>
</tr>
<tr>
<td>Bi-LSTM + CRF (J)</td>
<td>53.83</td>
<td>79.43</td>
<td>57.51</td>
</tr>
<tr>
<td>mBERT (I)</td>
<td>54.17</td>
<td>78.81</td>
<td>57.61</td>
</tr>
<tr>
<td>mBERT (J)</td>
<td>54.20</td>
<td>79.12</td>
<td>57.71</td>
</tr>
<tr>
<td>pre-process + mBERT + CNN (I)</td>
<td>54.83</td>
<td>79.85</td>
<td>57.88</td>
</tr>
<tr>
<td>pre-process + mBERT + CNN (J)</td>
<td>54.92</td>
<td>80.23</td>
<td>57.90</td>
</tr>
</tbody>
</table>

**Table 4:** Macro-F1 and Accuracy metric for Fine-grained (FN) POS tagging on CM social media data (Be stands for Bengali, En for English, Hi for Hindi, Te for Telugu)

<table>
<thead>
<tr>
<th>POS (FN)</th>
<th>Be-En</th>
<th>Hi-En</th>
<th>Te-En</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>macro-F1</td>
<td>Acc</td>
<td>macro-F1</td>
</tr>
<tr>
<td>CRF (I)</td>
<td>45.24</td>
<td>56.19</td>
<td>45.69</td>
</tr>
<tr>
<td>CRF (J)</td>
<td>45.63</td>
<td>68.17</td>
<td>47.17</td>
</tr>
<tr>
<td>Bi-LSTM + CRF (I)</td>
<td>47.24</td>
<td>79.97</td>
<td>48.63</td>
</tr>
<tr>
<td>Bi-LSTM + CRF (J)</td>
<td>47.23</td>
<td>79.43</td>
<td>49.51</td>
</tr>
<tr>
<td>mBERT (I)</td>
<td>48.43</td>
<td>78.21</td>
<td>50.13</td>
</tr>
<tr>
<td>mBERT (J)</td>
<td>48.45</td>
<td>78.47</td>
<td>50.48</td>
</tr>
<tr>
<td>pre-process + mBERT + CNN (I)</td>
<td>48.51</td>
<td>78.41</td>
<td>50.17</td>
</tr>
<tr>
<td>pre-process + mBERT + CNN (J)</td>
<td>48.47</td>
<td>78.67</td>
<td>50.78</td>
</tr>
</tbody>
</table>
tion due to such diverse learning scope.

Recently meta embedding representations that include both the pre-trained embeddings and domain-specific fine-tuned embeddings are achieving great results in the field of NLP. The joint learning with meta embeddings is left as future scope.

References


Sunil Gundapu and Radhika Mamidi. 2018. Word level language identification in english telugu code mixed data. In PACLIC.


Abstract

Previous research has used linguistic features to show that translations exhibit traces of source language interference and that phylogenetic trees between languages can be reconstructed from the results of translations into the same language. Recent research has shown that instances of translationese (source language interference) can even be detected in embedding spaces, comparing embedding spaces of original language data with embedding spaces resulting from translations into the same language, using a simple Eigenvector-based divergence from isomorphism measure. To date, it remains an open question whether alternative graph-isomorphism measures can produce better results. In this paper, we (i) explore Gromov-Hausdorff distance, (ii) present a novel spectral version of the Eigenvector-based method, and (iii) evaluate all approaches against a broad linguistic typological database (URIEL). We show that language distances resulting from our spectral isomorphism approaches can reproduce genetic trees on a par with previous work without requiring any explicit linguistic information and that the results can be extended to non-Indo-European languages. Finally, we show that the methods are robust under a variety of modeling conditions.

1 Introduction

The study of cross-linguistic variation has been a key focus of linguistics for genetic or typological classification of languages. Historical comparative linguistic methods determine genetic relationships between languages using concept lists of words with a common origin in multiple languages that share similar meaning and pronunciation (Swadesh, 1952; Dyen et al., 1992). Linguistic typology studies how distinct languages are, and what generalizations can be made regarding cross-linguistic variation on different levels of linguistic analysis and representations (Trask, 2000). Comrie (1989), for example, studies language variance in terms of their functional processes, whereas Cysouw (2013) measures language distance using structural features. More recent research indicates that semantic similarity between languages can serve as a quantitative means to determine cross-linguistic variation across languages. Seminal work of Eger et al. (2016) provides evidence that semantic alignment between languages can be explained by geographical factors. Likewise, Thompson et al. (2018) find that differences correlate with cultural distances among societies speaking the languages.

Conversely, it has also been shown that language differences are so profound that the structure of a language is approximately preserved even when translated into another language. This is often referred to as source language interference (Toury, 2012). Rabinovich et al. (2017) show that source languages of translations into the same target language can be clustered solely based on interference phenomena in the translations in the target language using simple linguistic features and that these clusters correspond with genetic distance. In a similar vein, Bjerva et al. (2019) find that comparable results can be established by clustering neural language model (NLM) based vectors using raw words, part-of-speech (POS) tags, phrase-structure or dependency-based input sequence representations of the data, showing that the distances between these learned language representations are more reflective of syntactic (structural) similarity rather than genetic relationship. Chowdhury et al. (2020) show that source language interference is even evident in simple word, POS, synset and semantic tag based embedding spaces computed from originally authored and data translated into the same target language. They use a graph-based Eigenvector (EV) divergence from iso-
morphism distance measure (Søgaard et al., 2018), originally used for bilingual dictionary induction, to capture divergence from isomorphism between monolingual original and translation embedding spaces. With this, they quantify distances between the source languages of the translations and predict phylogenetic trees, and analyse the correlation between isomorphism measure based distances and genetic relations in language families.

However, to date, (i) alternative graph-based distance metrics have not yet been explored for embedding-based approaches to detect translationese; (ii) it is not clear how word-embedding based approaches fare under different data settings including a) varying the number of most frequent words considered in the graphs, b) different corpus sizes and c) different word embedding architectures; (iii) it is not clear how the previous approaches (using either linguistic feature vectors, NLM based feature vectors, or divergence from isomorphism graph-based distance between embedding spaces) compare on the same data against the commonly used gold standard phylogenetic tree of Serva and Petroni (2008) (SP08); (iv) it is not clear how function words would affect graph-based distances; (v) evaluation of the graph- and embedding-space approach against the broader URIEL typological data base (Littell et al., 2017) has not been carried out; and (vi) it is not clear if the scope of this research can be expanded to include non-Indo-European languages.

In this paper, we show (i) that Gromov-Hausdorff (GH) distance can be used as a distance metric to quantify divergence from isomorphism between simple embedding spaces in monolingual settings and develop a novel Spectral Graph-based (SGM) distance measure, extending the original EV-based approach; (ii) that graph- and embedding-based distances are fairly robust under different data settings and that they are not sensitive to skip-gram or CBOW-based embeddings; (iii) divergence from isomorphism graph-based measures using embeddings can reproduce genetic trees on a pair with linguistic feature vector and NLM based approaches (Rabinovich et al., 2017; Bjerva et al., 2019); (iv) that function words and concept lists are still relevant within this general approach; (v) that graph- and embedding space-based distance metrics correlate not only with genetic features but also with geographical and syntactic ones (Littell et al., 2017); and (vi) that this research can be extended to translations from non-Indo-European languages.

The rest of the paper is organised as follows. We review related work in Section 2. Section 3 introduces the concept of graph isomorphism and our SGM measure, together with EV and GH. We describe the experimental setting in Section 4. In Section 5, we report results on the isomorphism metrics, infer language family relationships and correlate them with linguistic benchmarks. We describe robustness experiments in Section 6 and compare to previous work in Section 7. Finally, we extend our analysis to non-Indo-European source languages in Section 8 and summarize and draw conclusions in Section 9.

2 Related Work

Representational distance between two languages refers to how different one language or language variety is from another. Several analyses (Malaviya et al., 2017; Onecevay et al., 2020) have attempted to disentangle the typological factors that influence language representational distance. Rabinovich et al. (2017) clustered languages based on linguistically inspired features of their translations into the same target language and show that syntactic footprints of the source language in the translations can be used to estimate phylogenetic similarities between their source languages. They use agglomerative clustering with variance minimization (Ward Jr, 1963) as linkage procedure and compare their generated trees (P) to the pruned gold-tree (g) (SP08) of Serva and Petroni (2008). Their comparison metric is the sum of squared deviations between each language pair’s gold-tree distance and corresponding distance in their computed tree DP:

\[ \text{Dist}(P, g) = \sum_{i,j} (D_P(l_i, l_j) - D_g(l_i, l_j))^2 \] (1)

It is worth noting that the use of SP08 as a gold standard has also been questioned in the literature. Fortson IV (2011) observes that the SP08 approximation is only suited for a small subset of languages and that it fails to explain finer-grained inconsistencies in the Indo-European language family.

Bjerva et al. (2019) expand the work of Rabinovich et al. (2017) in their NLM- and sequence-based approach and argue that representational distance between languages can be better explained by structural relatedness than by language genetics. Chowdhury et al. (2020) use departures from
isomorphism based on the EV measure (Søgaard et al., 2018) on simple embeddings to infer genealogical distances. They compare different embedding spaces (word, POS, synset or semantic tags) constructed from translations into a single target language and the target language in terms of how similar their corresponding nearest neighborhood graphs are by analyzing their eigenvalues.

The similarity between languages can also be measured using the (dis)similarities between their discrete linguistic properties. Such properties are typically handcrafted and collected in typological databases such as URIEL (Littell et al., 2017) which lists a large inventory of properties for 8000 languages of various typological characteristics, such as overlap in syntactic features, or proximity along phoneme features (Cysouw, 2013). URIEL is a compilation of a variety of linguistic resources including the World Atlas of Language Structure, WALS (Dryer, 2009), PHOIBLE (Moran et al., 2014), Ethnologue (Lewis et al., 2015), and Glotolog (Nordhoff and Hammarström, 2011). Based on linguistic feature vectors, URIEL provides precomputed distance statistics between any language pairs stored in the database in terms of various metrics including genetic, geographical, syntactic, phonological, and phonetic inventory distances.

In this work, we follow the approach of Rabinovich et al. (2017) and evaluate our geometrical measures against the phylogenetic benchmark SP08. We compute the branching length directly from SP08, assuming it reflects the actual proportions. Additionally, we follow He et al. (2019) to compare our generated trees against the average of three precomputed measures of language distance, namely genetic, geographic, and syntactic distances based on the URIEL database.

3 Graph Isomorphism

We define the distance between languages based on word usage and the notion of isomorphism. An isomorphism \( f \) between two metric spaces \((\mathcal{X}, d_\mathcal{X})\) and \((\mathcal{Y}, d_\mathcal{Y})\), where \( \mathcal{X} \) and \( \mathcal{Y} \) are two sets of words in two languages and \( d_\mathcal{X} \) and \( d_\mathcal{Y} \) are the metric distances, is a function \( f : \mathcal{X} \mapsto \mathcal{Y} \) that is a distance preserving transformation i.e.: for all pairs of points \( x_1 \) and \( x_2 \) in \( \mathcal{X} \) such that, \( d_\mathcal{Y}(f(x_1), f(x_2)) = d_\mathcal{X}(x_1, x_2) \).

For a vocabulary \( V = v_0, v_1, ..., v_n \) in language \( \ell \), we define its graph as \( G(V, E, w) \), where \( V \) denotes the set of vertices corresponding to the vocabulary words; \( E = e_0, e_1, ..., e_m \) is a set of edges; and every pair \( \{v_i, v_j\} \) has a non-negative edge weight \( w_{ij} \) associated with it. Our approach starts with mapping words \( v_\ell^i \) in language \( \ell \) onto points \( v_\ell^i \) using distributional semantics methods. Each language is then represented with its own graph \( G_\ell^i \). After mapping words onto points \( v_\ell^i \) as vectors, the distance between words is defined as the distance between their vectors. We quantify the similarity between languages \( \ell_1 \) and \( \ell_2 \) through a distance function between their graphs \( d(G_\ell^1, G_\ell^2) \). In what follows we make the concept mentioned above more concrete.

3.1 Gromov-Hausdorff (GH) Distance

The first measure we use to quantify the similarity between languages is the Gromov-Hausdorff distance (GH) proposed by Patra et al. (2019).

Given two metric spaces \((\mathcal{X}, d_\mathcal{X})\) and \((\mathcal{Y}, d_\mathcal{Y})\), we start with the Hausdorff distance, defined as:

\[
d_{H}(\mathcal{X}, \mathcal{Y}) = \max \left\{ \sup_{x \in \mathcal{X}} d(x, \mathcal{Y}), \sup_{y \in \mathcal{Y}} d(y, \mathcal{X}) \right\}
\]

where \( d(a, \mathcal{B}) = \inf_{b \in \mathcal{B}} ||a - b||_2 \) is the distance of point \( a \) in \( \mathcal{A} \) from set \( \mathcal{B} \). Informally, it is the largest distance needed to travel from a point in \( \mathcal{A} \) to a point in \( \mathcal{B} \).

However, the Hausdorff distance is easily affected by isometric transformations. The GH distance which is the infimum of the Hausdorff distances under all possible isometric transformations is a more robust measure. By contrast, the GH distance reduces the distance over the isometric transforms \( f \) and \( g \) between \( \mathcal{X} \) and \( \mathcal{Y} \) as follows:

\[
d_{GH}(\mathcal{X}, \mathcal{Y}) = \inf_{f,g} d_{H}(f(\mathcal{X}), g(\mathcal{Y}))
\]

The computation of Hausdorff distance is NP-hard, and hence we follow Patra et al. (2019) and compute the Bottleneck distances (Chazal et al., 2009) which are considered to be reasonable lower-bounds.

3.2 Spectral Graph-based Matching (SGM)

Our second measure is based on the graph-based Eigenvector similarity method. Søgaard et al. (2018) used this similarity to measure the distance between two embedding matrices corresponding to two languages \( \ell_1 \) and \( \ell_2 \), via their Laplacian matrices, \( L \). They argue that the Laplacian eigenvalues are good compact representations for the
graph Laplacian, and that their comparison can consequently capture the degree of isomorphism. Although similar in spirit to their approach, our method to build the underlying graphs \((G_{\ell}^1\text{ and } G_{\ell}^2)\) differs. We use the same idea to model differences between two embedding spaces \(X\) and \(Y\) for the single target language translations from different source languages, as proposed by (Søgaard et al., 2018) but our method to build the underlying graphs \((G_{\ell}^1\text{ and } G_{\ell}^2)\) differs from their approach. While they extract the nearest neighbors by computing the cosine similarity of the cross-lingual word pairs, we take inspiration from the Isomap algorithm of Tenenbaum et al. (2000) and build a weighted connected graph over the data points to capture better neighborhood relations. Weights \(w_{ij}\) correspond to the distance between points \(i\) and \(j\) in the input space \((X, d_X(i, j))\). We connect each point only to its \(K\) nearest neighbors to consider more geometrical information on the interaction between all vectors within the initial space to improve the graph characterization of the spaces. The value \(K=6\) is chosen to have similar edge density for all graphs. We estimate the geodesic distances between vertices (points) in the input space using shortest-path distances obtained with Dijkstra’s algorithm (Dijkstra et al., 1959) on the constructed graph to minimize the sum of the weights of their constituent edges. The subsequent distance matrix represents the basis for our graphs. From this point onwards, the computation of the Laplacian matrices and the final measure \(\Delta\) is as in Søgaard et al. (2018), where

\[
\Delta = \sum_{i=1}^{k} (\lambda_{1i} - \lambda_{2i})^2. \tag{4}
\]

First for \(L_1\), we find the smallest \(k\) in Equation 4 such that the sum of its \(k\) largest eigenvalues \(\sum_{i=1}^{k} \lambda_{1i}\) is at least 90% of the sum of all its eigenvalues. Similarly, we find another \(k\) for \(L_2\), and take the smallest of these two, such that, \(k = \min(k_1, k_2)\). The graph similarity metric returns a value in the half-open interval \([0, \infty)\), where values closer to zero indicate better isometry. We compare our SGM metric with the metric of Søgaard et al. (2018) (referred to as EV) in Section 5.

4 Experimental Setting

In this section, we provide information on the data, and the vector spaces used for computing deviation of isomorphism. Since we quantify language similarity based on the degree of isomorphism in monolingual spaces, we independently train monolingual word embeddings for the target language and translations into that target language.

4.1 Data

We use the same setup as Rabinovich et al. (2017), Bjerva et al. (2019) and Chowdhury et al. (2020), and use the comparable portion of Europarl (Koehn, 2005) with translations from 21 European Union languages into English to minimize the impact of domain difference. The tokens per language vary, ranging from 67k tokens for Maltese to 7.2M for German. We refer to the multiple translations into English as \(L_j\)’s, where \(j=1,2,...,n\); and to originally written text in English as \(L_o\).

We select the subset of translations from 16 languages covering four families: Romance (French (fr), Italian (it), Spanish (es), Romanian (ro), Portuguese (pt)), Germanic (Dutch (nl), German (de), Swedish (sv), Danish (da), Slavic (Czech (cs), Slovak (sk), Slovenian (sl), Polish (pl), Bulgarian (bg)) and Baltic (Latvian (lv) and Lithuanian (li)) into English and English original (en) text.

For these 17 datasets, we define two settings, the full data condition and the small data condition to investigate the effect of data settings on our methods. The former makes use of the complete Europarl edition available for a language (recall that data size differs widely); for the latter, we randomly extract \(m\) sentences, where \(m\) corresponds to the lower-bound data-size of our translationese data, i.e., the size of the Latvian corpus (118,525 words). We shuffle and randomly subsample \(m\) sentences with the same seed for all target-side language data. We report results for the full data setting and use the small data for robustness checks and comparisons with existing literature.

4.2 Vector Spaces

Our data are original English \((L_o)\) or translations from language \(j\) into English \((L_j)\). For each data set we induce separate monolingual embeddings in the full and small data conditions, from their respective tokenised (Koehn et al., 2007) and lower-cased data using fastText (Bojanowski et al., 2017).

We train 300 dimensional embeddings with words with more than 5 occurrences in the data. We use skip-gram with negative sampling (Mikolov et al., 2013) with standard hyper-parameters (character \(n\)-grams of sizes 3 to 6, and a learning rate of...
Afterwards, embeddings are mean centered and unit normalised. For comparison purposes, we also create vector spaces using the CBOW algorithm and standard hyper-parameters.

5 Results and Evaluation

We analyze the behaviour of the different distance measures: the Gromov-Hausdorff distance (GH) and the two Eigenvector similarity-based ones (SGM and EV). We apply them to (i) infer language families, (ii) reconstruct phylogenetic trees, and finally (iii) perform correlation analysis against two benchmarks (SP08 and URIEL).

5.1 Language Distance Measures

First, we perform an experiment to determine how distant the vector spaces created from $L_o$ and $L_j$’s are. We compute each metric over the top-$n$ most frequent common words in our data, where $n \in \{1000, 1500, 2500, 3500\}$ to explore the behaviour of the measures with different graph sizes. Notice that having the same number of components (vertices and edges) is a condition for isomorphism.

Results with the normalised distances are displayed in Figure 1. The behaviour for the metrics varies with respect to the number of the most frequent top-$n$ datapoints considered. SGM is the most stable measure across all configurations, showing most variance for 1000 points. EV shows larger variability in distinguishing language similarity, while GH results are relatively stable with respect to larger datapoints (2500 and 3500 points). These variations are due to the different nature of the metrics. GH calculates the distance between the spaces only on the subset of $n$ words, while SGM weights the nearest neighbors of a word which could lie outside the top-$n$ to build the initial graph. Thus SGM considers more context for each point and thus needs less datapoints to successfully describe the space. As expected, results with EV and SGM are closer to each other than to GH because they follow a similar methodology, except that SGM retains more context than EV.

We observe that, in all plots, the differences between original English and translations from Germanic, followed by Romance languages are the lowest, indicating that vector spaces of these languages are closer to each other in terms of semantic embedding space based isomorphism measures. However, isomorphism weakens consistently with increased linguistic distances of Baltic and Slavic families irrespective of the method used, providing evidence that language distance in semantic space is higher for etymologically distant language pairs. Additionally, we observe some outliers varying from measure to measure. GH puts ro far from the other Romance languages and da and sv are placed relatively far from other Germanic relatives.
On the other hand, SGM (and to lesser extent EV) locates pl close to en.

5.2 Reconstructing Language Phylogeny

Figure 2 shows evidence that deviations from isomorphism between semantic spaces computed from $L_j$'s into a common target language (en) and originally authored text ($L_o$) in en reflect linguistic notions of distance between the source languages and the source language families of the translations. This is evidence for an important aspect of translationese, namely source language interference, in semantic space. Below we further investigate whether the distance in semantic space signal can be used to infer phylogenetic trees.

In our predicted trees, Figure 2, we observe trends that indicate groupings based on morphological or other typological properties. We identify some well known language–language relationships in all three trees showing high similarity between English and other Germanic languages, with some divergences —for example, sv is located far away from its other Germanic counterparts under GH reconstructions and pl is always misplaced into the Germanic-Romance language group, despite its Slavic origin. The influence of geographical factors such as language contact or structural interactions (Balkan Sprachbund) can also explain some of the interesting divergences. Overall, the trees exhibit coarse-grained language family contour traces, i.e., Baltic and Slavic languages are close together, while Germanic and Romance form another group in most of the cases.

Our simple embedding-based results provide evidence that translationese is reflected in semantic spaces and that without reliance on fine-grained linguistic knowledge, differences in semantic embeddings space are powerful enough to detect important language differences related to linguistic typology in semantic spaces, corroborating previous results of Chowdhury et al. (2020). This further reconfirms in word embedding based semantic space earlier findings of Rabinovich et al. (2017), Bjerva et al. (2019) which used manual feature engineering or NLMs with a focus on morphologic and syntactic structure.

5.3 Correlation with Typology, Geography and Phylogeny Benchmarks

Above we observed how predicted trees not only show genetic effects, but also other characteristics that might be due to the geographic proximity and not to just to phylogenetic evolution. In this section, we compare our language classification predictions, Figure 2, against linguistic benchmarks. We estimate Kendall correlations between our generated trees and SP08 (representing genetic similarities), and our trees and the averaged URIEL features introduced in Section 2 (representing other rich typological similarities beside genetic ones). The Kendall correlation between the two benchmarks SP08 and the selection of URIEL features is 0.56 reflecting the different nature of the two benchmarks. Although the genealogical distance is common in both, the source for this kind of information is different.

Our results, summarised in the top rows of Table 1, show that correlations between predicted trees and URIEL are higher than with SP08, demonstrating that other factors besides genetics are reflected in the semantic spaces. SGM reproduces the genetic SP08 benchmark better than EV and GH, while GH clearly correlates better with structural URIEL features, followed by SGM and EV. This corroborates NLM-based findings of Bjerva et al. (2019) in our semantic word embedding based spaces: the differences and similarities between languages and language representations go beyond genetic (dis)similarities. Further, we find that while correlations are better under full data conditions, they exhibit a similar behaviour in simulated small-data scenarios, suggesting that our graph-based approaches are effective in a variety of data settings.

6 Robustness Analysis

After showing that exploiting departures from isomorphism between spaces can be used to predict relations between languages, we analyze the impact of various modeling assumptions and different training conditions that might have an effect in skewing the results.

6.1 Data Size Effects

Large differences in data sizes between high and low-resource languages have played a pivotal role in the performance of monolingual embeddings (Vulić et al., 2020; Sahlgren and Lenci, 2016). In our work, to some extent this is already minimized

---

1The clusters are computed over 3500 datapoints for each metric.
Figure 2: Clustering based on the distance matrix obtained for GH (left), EV (middle) and SGM (right).

<table>
<thead>
<tr>
<th># Points</th>
<th>SP08</th>
<th>URIEL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GH</td>
<td>SGM</td>
</tr>
<tr>
<td>Full data condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>0.32</td>
<td>0.52</td>
</tr>
<tr>
<td>1500</td>
<td>0.40</td>
<td>0.30</td>
</tr>
<tr>
<td>2500</td>
<td>0.42</td>
<td>0.38</td>
</tr>
<tr>
<td>3500</td>
<td>0.40</td>
<td>0.39</td>
</tr>
<tr>
<td>FW</td>
<td>0.40</td>
<td>0.30</td>
</tr>
<tr>
<td>Swadesh</td>
<td>0.30</td>
<td>0.32</td>
</tr>
<tr>
<td>Small data condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>0.11</td>
<td>0.45</td>
</tr>
<tr>
<td>1500</td>
<td>0.29</td>
<td>0.37</td>
</tr>
<tr>
<td>2500</td>
<td>0.39</td>
<td>0.45</td>
</tr>
<tr>
<td>3500</td>
<td>0.27</td>
<td>0.46</td>
</tr>
</tbody>
</table>

Table 1: Mean Kendall correlations of predicted trees with SP08 and average URIEL for various number of datapoints and the function words experiment (FW).

by taking only the most frequent \( n \) words to estimate the distances between embedding spaces, but still the quality of even these embeddings might differ. To examine the impact of the data size for our experiments, we use the embeddings obtained under the small data condition (see Section 4) and compare the results in the bottom rows of Table 1.

The results show that SGM correlates best with SP08 under all training conditions (number of datapoints and corpus size), but the correlation decreases with respect to URIEL features. GH shows good correlation in some instances (1500 and 2500 datapoints) for both SP08 and URIEL, while EV, shows no consistent correlation. For EV, we consider frequent words and mutual nearest neighbors, thus in the small data condition, it has even less access to contexts. Our spectral graph-based measure SGM, which is inspired by the ideas of node representation in contemporary geometric and manifold learning (Cayton, 2005), provides more intuitive understanding of linguistic distances than what is offered in Chowdhury et al. (2020) under varied data settings.

6.2 Word Embedding Effects

Köh (2015) showed that different methods to obtain word embeddings (CCA, skip-gram, CBOW, GloVe, etc.) behave similarly when capturing syntactic and morphological information. We check that this is also the case with our distance methods by comparing the performance obtained with skip-gram and CBOW architectures, and observe small variations with similar global trends. To give an example, correlation results for SGM under the full data condition with CBOW and skip-gram vary only in the ±0.05 range.

We also performed experiments with lower dimensions (50,100,200) which may lead to reduced expressivity, but, very interestingly, we obtained similar performance as we did with 300 dimensions. For example, on 100-dimensional monolingual word embeddings, the differences are: GH (±0.077), SGM (±0.013), and EV (±0.088).

7 Comparison with Previous Approaches

7.1 Leaf-Node Tree Distances

In order to compare our results with previous work, we calculate tree distances using the leaf-node distance in Equation 1 previously defined in Rabinovich et al. (2017), and compare with the best results on SP08 in Rabinovich et al. (2017) and Bjerva et al. (2019). We report our results in the small and the large data conditions for 1500 most frequent datapoints obtained with different metrics in Table 2. All distances are normalized to a zero-one scale.\(^3\)

\(^3\)Notice that the results of Table 1 and Table 2 cannot be directly compared as Table 2 is computed after summing over all possible pairs of the leaves (languages), while Table 1 shows the association with benchmarks (SP08 and URIEL) keeping only originally authored English as its source. Table 1 follows Chowdhury et al. (2020) to correlate the results with the benchmarks and Table 2 compares the findings to Rabinovich et al. (2017); Bjerva et al. (2019).

\(^4\)Although an overall correlation similarity analysis based on confusion matrices would be more optimal, we perform...
According to the mean distance, our simple embedding and graph-based approaches, especially GH can reproduce genetic trees on a par with previous work without requiring any explicit linguistic information.

Unlike previous methods which rely on surface-level features of the source language, our graph-based isomorphism analysis is unsupervised and still is able to detect important language differences related to linguistic distances. Of all the methods, GH is the closest to SP08, followed by SGM and EV in the full data settings while the trend for SGM-EV is reversed under the small data condition.

7.2 Function and Content Words

To control topical skew, we investigate whether our approaches of departure from isomorphism works on non-lexical representations. To this end, we first focus on function words which introduce and identify key discourse referents and represent relationships between entities but are considered to be not well-modeled by distributional semantics (Bernardi et al., 2015). We use the list of function words defined in Koppel and Ordan (2011) to construct the language distance measure of Section 5.1 in Figure 3(a). In this case, the number of data points is 468, well below the minimum number of points used with content words (1000).

The performance of all three methods show similar trends as in Figure 1. The figure demonstrates that function words are able to capture departures of isomorphism in a similar way as the complete set of words, indicating that source languages carry over grammatical constructs into the translation product, corroborating in simple embedding space prior findings of Rabinovich et al. (2017) and Bjerva et al. (2019) with function words.

Additionally, we explore the much smaller cog-
nate collection of Swadesh word lists (Swadesh, 1952) to capture the relatedness between languages in Table 1 and the language distance computed from their embeddings is shown in Figure 3(b). As this concept-aligned resource ensures a consistent set of word-lists across all our languages, thereby enhancing comparability, these findings are particularly important. The results in Table 1 show that the large context (6 neighbors) exploited by SGM estimations exceeds other isomorphism methods, while highlighting the limitations of EV in low-data regimes with limited access to contexts.

8 Analysis for non-Indo-European Source Languages

Previous research (Rabinovich et al., 2017; Bjerva et al., 2019; Chowdhury et al., 2020) focused on investigating translationese and source language interference for European language families. Here we extend this work, for the first time, to the best of our knowledge, to translations from non-Indo-European languages into English. We explore the language distance measures of Section 5.1 on the UN corpus (Tolochinsky et al., 2018) which consists of translations covering typologically different languages such as Arabic (ar) and Chinese (zh), as well as Indo-European languages (i.e., Russian (ru), Spanish (es) and French (fr)). The embedding spaces are created in the same manner as for Europarl dataset.

We show results with 1500 words in Figure 4 and observe the following trends: compared with translations from ar and zh, the difference between original English and translations from fr and es tend to be smaller, the distance to zh is the largest, and that within the European language family distances is mostly \( fr < es < ru \). This is in line with our previous results in Figure 1 on the same-domain monolingual Europarl data under different data settings. However, despite these general trends, GH and EV measured distance scores are similar for es, ar and ru, while EV has fr more distant to en than es, ru and ar. This is something that would need to be further explored in future work. Of all measures, SGM, which captures more context from the interaction between data-points and their neighborhoods, accords best with linguistic expectations about language (dis)similarities.

9 Conclusion

In this paper we contribute to the ongoing line of research in computational typology, exploring the potential of translations into a single target language that retains the traces of the source languages to reflect the distances between them. Specifically, we propose an alternative graph-based distance measure to explore (dis)similarities between languages. Our results show that simple graph- and embedding-based distance based methods perform on a par with the best results achieved by previous approaches based on linguistic features in detecting source language interference in translations. We compare Gromov-Hausdorff and our novel Spectral Graph based approach with the original Eigenvector-based divergence from isomorphism measure (EV) against URIEL and SP08, show that our alternative graph isomorphism measures outperform EV and, for the first time, expand translationese research to non-Indo-European languages. We perform robustness tests to verify that our methods are stable under a variety of modeling conditions.

In future work, we aim to leverage our estimated similarities to better explain transfer behavior (local information spreading from one language to the other) in downstream applications such as machine translation.

Acknowledgments

We would like to thank the reviewers for their insightful comments and feedback. This research is funded by the German Research Foundation (Deutsche Forschungsgemeinschaft) under grant SFB 1102: Information Density and Linguistic Encoding.

References


Steven Moran, Daniel McCloy, and Richard Wright. 2014. PHOIBLE online.


Abstract

Large transformer models, such as BERT, achieve state-of-the-art results in machine reading comprehension (MRC) for open-domain question answering (QA). However, transformers have a high computational cost for inference which makes them hard to apply to online QA systems for applications like voice assistants. To reduce computational cost and latency, we propose decoupling the transformer MRC model into input-component and cross-component. The decoupling allows for part of the representation computation to be performed offline and cached for online use. To retain the decoupled transformer accuracy, we devised a knowledge distillation objective from a standard transformer model. Moreover, we introduce learned representation compression layers which help reduce by four times the storage requirement for the cache. In experiments on the SQUAD 2.0 dataset, a decoupled transformer reduces the computational cost and latency of open-domain MRC by 30-40% with only 1.2 points worse F1-score compared to a standard transformer.

1 Introduction

Open-domain question answering (QA) aims to answer questions from a collection of text passages. It is an important and challenging task with application to several domains such as web search and voice assistants. The most popular architecture for open-domain QA is retriever-reader (Chen et al., 2017). Given a question, the retriever uses an information retrieval (IR) system over a collection of passages to return top-K results that are most likely to contain an answer. Then, the reader uses a machine reading comprehension (MRC) model on each of the top-K results to find an answer. In the end, the top-K MRC answers are ranked to produce a final answer.

For both the retriever and the reader, large transformer models such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019) and ELECTRA (Clark et al., 2020) achieve state-of-the-art results. A disadvantage of large transformer models is the high computational cost for inference which makes them hard to apply to online runtime systems, e.g. voice-assistants. Transformers’ computational cost comes from three major factors. Firstly, the size of the feed-forward layers which project to an intermediate higher dimension and projects back to the original dimension. Secondly, the multi-head self-attention has quadratic computational complexity in the sequence length. Thirdly, the total number of layers.

Dense passage retrieval (DPR) (Karpukhin et al., 2020) is a retriever model which uses a transformer question encoder and transformer passage encoder to capture semantic similarity. The question and passage encoders are trained such that passages that are likely to contain an answer have a large embedding dot product with the question embedding. The embeddings for the passages are generated offline and indexed for efficient distributed KNN search (Johnson et al., 2017), and only the embedding for the question is generated at runtime. Since questions are usually short, the retriever runtime inference computational cost is low.

MRC reader models process the top-K passages returned by the retriever to get an answer. In transformer-based MRC models, each passage is encoded together with the question using a CLS and separator characters [CLS] Question [SEP] Passage. The encoding is followed by a prediction head which determines the answer span. If there is no answer, the model result is a zero-length span on the CLS token. The joint encoding of the document and the question produces rich interaction features but it increases the sequence length, and thus the computational coast. Since the
MRC model inference is executed at runtime on long sequences of a question and passages, MRC is the main computational bottleneck in retriever-reader QA.

There have been several ideas to reduce the runtime inference of transformer models, such as precision reduction via quantization (Zafrir et al., 2019; Shen et al., 2020), knowledge distilling to a smaller architecture (Sanh et al., 2019; Jiao et al., 2019), and approximate multi-head attention for reducing the quadratic complexity (Wang et al., 2020; Beltagy et al., 2020). In this paper, we take an orthogonal approach, decoupling the transformer encoding for multiple inputs to improve efficiency, which can be combined with the aforementioned techniques. The motivation for the decoupled transformer is that in open-domain QA the passages are known in advance and part of the passage computation can be performed offline and stored. Then, online at the runtime the question computation can be performed only once and combined with the stored state from passages with cross-attention.

We use the decoupled transformer to reduce the computational cost of open-domain MRC by 30-40% with only 1.2 points worse than the F1-score on the SQUAD 2.0 benchmark.

Our contributions are as follows:

• We propose and evaluate a novel decoupled transformer approach for MRC in open-domain QA to reduce runtime inference cost. Our approach uses a knowledge distillation (KD) objective to bridge the gap between a standard transformer and decoupled transformer.

• We conduct experiments to understand how much cross-attention between inputs is needed in MRC and other natural language processing (NLP) tasks like paraphrasing identification and natural language inference.

• We devise an accurate representation compression approach to reduce the storage requirement for decoupled transformer offline state. The compression provides a four-fold reduction in the index storage requirement for large corpora such as Wikipedia, from 3.4 TB to 858 GB.

2 Related Work

DC-BERT (Zhang et al., 2020) is a decoupled transformer that has dual BERT models. An online BERT encodes the question and an offline BERT pre-encodes all the passages and stores their encodings. Conceptually, DC-BERT goals and architecture of combining local and global context are similar to our work with the following major differences:

• We apply decoupled transformers to MRC and DC-BERT is designed and evaluated for the retrieval passage ranking. With the recent introduction of DPR, passage ranking as explored in DC-BERT is less important, so MRC becomes the primary bottleneck.

• We investigate how much cross-attention is needed for MRC and other NLP tasks.

• We introduce compression and decompression layers to reduce representation storage requirements.

Another model where the query and the passage are encoded independently using a transformer is ColBERT (Khattab and Zaharia, 2020). The main modeling applications for ColBERT are retrieval and passage ranking. After encoding the query and the passage independently, late interactions are introduced using an efficient sum of maximum similarity computations. ColBERT representations are used for retrieval, so it combines the strengths of DPR and DC-BERT. However, the efficient late interactions in ColBERT do not have enough representation power for complex tasks like MRC.

3 Decoupled Transformer

In the decoupled transformer, Figure 1, we split the transformer model \( M \) into two components.

1. Input-component \( M_{\text{input}} \) (the lower \( N \) layers) which processes the inputs independently and produces a representation. The representation for the inputs that are known in advance, i.e. the passages, is stored and used without computation.

2. Cross-component \( M_{\text{cross}} \) (the higher \( M \) layers) which processes the inputs jointly (after concatenation) and produces the final output.

3.1 Workflow

The workflow is depicted in Figure 2. Offline, we run the input-component \( M_{\text{input}} \) on each passage from the collection of passages and store the representation in the search index. Moreover, we compress the stored passage representation to reduce
Figure 1: On the left, standard transformer model for MRC. On the right, decoupled transformer model with input-component and cross-component.

3.2 Benefits

The decoupled transformer reduces per question transformer complexity in lower $N$ layers from $O(N_p(L_q + L_p)^2)$ to $O(L_q^2 + N_pL_p^2)$ where $N_p$ denotes the number of top-K passages per question, $L_q$ and $L_p$ denote the average number of tokens of each question and passage.

At runtime, the computation for the lower $N$ layers for the passage is eliminated because it is performed once offline and reused. Moreover, the computation for the lower $N$ layers for the question is done only once for the top-K retrieved passages, and not repeated, as opposed to the normal transformer which uses all layers on both the question and the top-K retrieved passages.

3.3 Initialization

To build a decoupled transformer model, we start from a standard transformer such as BERT, RoBERTa, ELECTRA model which is fine-tuned on a target dataset such as SQUAD 2.0. Then, we create the decoupled transformer model by splitting the encoder layers into input and cross components which are initialized with the fine-tuned MRC model weights. In addition to the standard transformer weights, we create a global position embedding and segment embedding layers at the start of the cross-component and initialize them to the same weight as the local position and segment embedding from the input-component. The global position and segment embedding re-encode the tokens for the new position in the concate-
nated question-document encoding sequence. The segment embedding differentiates whether the encoded token is from the question or document.

### 3.4 Training Objective

During decoupled transformer training, we aim to preserve the standard transformer model accuracy. To achieve that, we propose a knowledge distillation (KD) (Hinton et al., 2015) objective from the standard transformer to decoupled transformer which helps preserve the original representation.

The objective function is the sum of four terms:

\[
L = (1 - \lambda)CE(y, \text{target}) \\
+ \lambda KL(\text{logits}/T, \text{teacher-logits}/T) \\
+ \sigma MSE(\text{represent}_n, \text{teacher-represent}_n) \\
+ \sigma MSE(\text{attention}_n, \text{teacher-attention}_n)
\]

1. A standard cross-entropy (CE) loss with the prediction \(y\) and hard targets from ground truth labels.
2. KD loss based on Kullback–Leibler (KL) divergence with logits from the teacher standard transformer model. We scale the targets with the same temperature \(T\) for both the teacher and student.
3. The mean square error (MSE) between the decoupled model final layer representation with the original model final layer representation.
4. The MSE between the decoupled model final layer multi-head self-attention output with the standard model final layer multi-head self-attention output.

The parameter \(\lambda\) determines the relative contribution of CE and KL losses. And, \(\sigma\) is a weight for the MSE losses.

The MSE losses on the final layer representation and the final layer self-attention are similar to TinyBERT (Jiao et al., 2019) approach to smaller model distillation. Unlike TinyBERT, we only apply the MSE losses only on the last layer. The motivation for the MSE losses is that we are aiming to make the representation at the end of the decoupled transformer to match the representation of the standard transformer.

### 4 Representation Compression

In open-domain QA the collection of passages are known in advance. So, with decoupled transformer, we run the input-component \(M_{\text{input}}\) on each passage offline and store the passage representation in the index. For a large corpus, the representation storage can be a significant amount. In the case of QA over Wikipedia, the storage requirement is around 3.4TB given around 32 million passages, averaging 150 tokens per passage, and 768 token dimensions for the BERT-base model with 16-bit precision.

To reduce the storage requirements for the passage representation of the decoupled transformer, we introduce a compression layer at the end of the input-component and a decompression layer at the start of the cross-component, see Figure 2. The compression layer is a linear projection from the original dimension to a compression dimension. The decompression layer is a linear projection from the compression dimension to the original dimension. These layers are similar to an autoencoder with a bottleneck.

### 4.1 Training Procedure

To train the compression and decompression layers we start from a decoupled transformer model. Then, we perform training in two phases:

- **Phase 1.** We train the randomly initialized compression and decompression layers to reconstruct the input-component output representation without updating the decoupled transformer model itself.
- **Phase 2.** We train the compression and decompression layers together with the decoupled transformer jointly. This means the cross-component receives the decompressed representation.

The intuition behind the two-phase approach is that since compression and decompression layers are randomly initialized, it is beneficial to first train the compression and decompression layers independent from the decoupled transformer to get near-optimal weights. Then, train the cross-component of the model to understand the slightly different decompressed representation.

### 5 Experiments and Results

#### 5.1 Datasets

We evaluate the decoupled transformer on SQUAD 2.0 (Rajpurkar et al., 2018) which is a popular MRC dataset over Wikipedia articles. In addition
to MRC, we evaluate models on the datasets below to understand how many cross-components layers are needed for tasks of different complexity and dataset size.

- **QQP** (Chen et al., 2018) and **MRPC** (Dolan and Brockett, 2005) datasets for paraphrasing identification. The task is given two sentences, recognizing if they are paraphrases or not.
- **MNLI** (Williams et al., 2018) dataset for natural language inference datasets. The task is given two sentences the “premise” and the “hypothesis”, to determine if the hypothesis entails, contradicts, or is neutral given the premise.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warmup steps</td>
<td>200</td>
</tr>
<tr>
<td>Learning Rate (LR)</td>
<td>5e-5</td>
</tr>
<tr>
<td>Layer-wise LR Multiplier</td>
<td>0.95</td>
</tr>
<tr>
<td>Batch size per GPU</td>
<td>32</td>
</tr>
<tr>
<td>Number of GPUs</td>
<td>2</td>
</tr>
<tr>
<td>Adam $\epsilon$, $\beta_1$, $\beta_2$</td>
<td>1e-6, 0.9, 0.999</td>
</tr>
<tr>
<td>Attention Dropout</td>
<td>0.1</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.1</td>
</tr>
<tr>
<td>Weight Decay</td>
<td>Linear</td>
</tr>
<tr>
<td>Gradient Clipping</td>
<td>3.0</td>
</tr>
<tr>
<td>Epochs</td>
<td>4</td>
</tr>
<tr>
<td>KD temperature $T$</td>
<td>3.0</td>
</tr>
<tr>
<td>Loss weight $\lambda$</td>
<td>0.95</td>
</tr>
<tr>
<td>Loss weight $\sigma$</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 2: Model training hyperparameters.

### 5.2 Setup

#### Models

We use ROaD-base (ElFadeel and Peshterliev, 2021) for the MRC experiments on SQUAD 2.0. ROaD is an ELECTRA model pretrained and distilled using multi-task learning. For the experiments on QQP, MRPC, and MNLI we use ELECTRA-base. All models are implemented in PyTorch and optimized using Adam (Kingma and Ba, 2014).

#### Hyperparameters

Table 2 shows the hyperparameters that we use for fine-tuning the standard transformer and training the decoupled transformer. We searched different values for the temperature $T$, and the weights $\lambda$ and $\sigma$. For $\lambda$, we experimented with 0.5, 0.7, 0.9, 0.95 values and we found that for decoupled transformer training a large $\lambda$ that biases towards the KL divergence objective work best. For $\sigma$, we experimented with 0.25, 0.5, 0.75, 1.0 values and we found that smaller values work better because otherwise the KL divergence objective is given less weight which leads to worse models.

#### Hardware

We perform the experiments and benchmarks on Nvidia Titan RTX with tensor cores GPU and AMD Ryzen Threadripper 3960X - 24 cores CPU.

### 5.3 Decoupled Transformer

First, we perform a set of experiments on a decoupled transformer without compression. For each experiment, we denote the decoupled transformer split configuration as $x$-$y$, where $x$ is the number of input-component layers, and $y$ is the number of cross-component layers.

Table 1 shows the performance and FLOPs starting from the baseline standard transformer model to decoupled transformer with extreme 11-1 split. We observe that tasks with a large dataset (QQP, SQUAD, MNLI contain over 100K samples each) have similar behavior with a noticeable drop when moving from decoupled transformer 5-7 split to 6-6 split, and another big drop when the number of cross-component layers becomes less than 3. While in MRPC, a small dataset with around 5K sample, the drop of performance was significant.

<table>
<thead>
<tr>
<th>Task</th>
<th>Baseline</th>
<th>1-11</th>
<th>2-10</th>
<th>3-9</th>
<th>4-8</th>
<th>5-7</th>
<th>6-6</th>
<th>7-5</th>
<th>8-4</th>
<th>9-3</th>
<th>10-2</th>
<th>11-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQUAD 2.0</td>
<td>87.6</td>
<td>87.5</td>
<td>87.2</td>
<td>87.1</td>
<td>87.0</td>
<td>86.7</td>
<td>85.4</td>
<td>85.2</td>
<td>84.8</td>
<td>84.0</td>
<td>80.6</td>
<td>62.6</td>
</tr>
<tr>
<td>QQP</td>
<td>91.5</td>
<td>91.1</td>
<td>91.0</td>
<td>90.9</td>
<td>90.9</td>
<td>90.9</td>
<td>90.5</td>
<td>90.4</td>
<td>90.4</td>
<td>90.0</td>
<td>89.0</td>
<td>86.4</td>
</tr>
<tr>
<td>MNLI</td>
<td>88.9</td>
<td>87.3</td>
<td>87.1</td>
<td>86.9</td>
<td>86.7</td>
<td>86.7</td>
<td>86.4</td>
<td>86.4</td>
<td>85.6</td>
<td>77.0</td>
<td>73.5</td>
<td></td>
</tr>
<tr>
<td>MRPC</td>
<td>89.5</td>
<td>87.7</td>
<td>86.3</td>
<td>85.5</td>
<td>83.0</td>
<td>80.1</td>
<td>78.1</td>
<td>77.9</td>
<td>77.8</td>
<td>77.7</td>
<td>71.8</td>
<td>71.6</td>
</tr>
<tr>
<td>FLOPs</td>
<td>1.0</td>
<td>.91</td>
<td>.83</td>
<td>.75</td>
<td>.66</td>
<td>.58</td>
<td>.50</td>
<td>.41</td>
<td>.33</td>
<td>.25</td>
<td>.16</td>
<td>.08</td>
</tr>
</tbody>
</table>

Table 1: Decoupled transformer results with variable number of input-component and cross-component layers. Baseline is a standard transformer model. The $x$-$y$ columns indicate the number of input-component and cross-component layers. We use F1 score for SQUAD 2.0 and accuracy for QQP, MNLI, and MRPC. There is a consistent trend for performance to degrade as we increase the number of input-component layers and decrease the number of cross-component layers. FLOPs is floating point operations for inference as a measure of computational cost.
and bigger than the other large datasets even with the decoupled transformer with 1-11 split.

With every layer, we moved from the cross-component to the input-component, the FLOPs decreased by about 8% and performance dropped by a small amount until the number of input-component layers equals to or bigger than the cross-component layers. The results show that choosing the right setting is application-specific and the best option depends on the particular performance and latency trade-offs.

For the following experiments, we use the 5-7 split because it provides the best trade-offs between accuracy and FLOPs across the evaluated datasets.

<table>
<thead>
<tr>
<th>MRC Model</th>
<th>SQUAD 2.0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Decoupled 5-7</td>
<td>F1</td>
<td>EM</td>
</tr>
<tr>
<td>- SQUAD 2.0 pretraining</td>
<td>86.7</td>
<td>84.1</td>
</tr>
<tr>
<td>- training position and segment embedding in the cross-model</td>
<td>84.2</td>
<td>81.5</td>
</tr>
<tr>
<td>- KL objective</td>
<td>82.1</td>
<td>80.0</td>
</tr>
<tr>
<td>- MSE on representation and attention final layer</td>
<td>84.0</td>
<td>80.4</td>
</tr>
<tr>
<td>+ MSE on hidden and attention applied to all layers</td>
<td>86.5</td>
<td>83.7</td>
</tr>
</tbody>
</table>

Table 3: Decoupled transformer ablation study for SQUAD 2.0 MRC decoupled transformer with 5-7 split. We remove one row at a time except for the last row where we add MSE losses to all layers and not just the final layer.

### Ablations

We perform an ablation study to understand the effect of the different modeling techniques on the decoupled transformer performance. Table 3 shows the results. First, we remove the SQUAD 2.0 pretraining and start with regular ELECTRA-base which reduces F1 significantly by 2.5 points. Then, we tried keeping the position and segment embeddings in the cross-component frozen which hurt F1 as expected. If we remove the distillation KL objective, F1 degrades significantly by 2.7 points. On the other hand, removing the MSE losses on the representation and attention does not cause a significant reduction in F1. However, adding MSE losses on all layers actually causes a reduction in F1 because the CE and KL objectives receive less weight.

### 5.4 Compression

To evaluate compression, we conducted experiments on a decoupled transformer with 5-7 split using the MRC model for SQUAD 2.0. Our goal is to understand how much impact different levels of compression have on the storage requirement and model performance.

Table 4 compares the results with five different levels of compression. We observed the performance degradation is minimal for 2x, 3x and 4x compression, and then it starts to degrade significantly. At 4x compression, the required storage for open-domain QA over Wikipedia with the previous assumptions is 3.4 TB which could be reduced to 858 GB.

**Ablations.** We evaluate the effectiveness of the two-stage training of the compression and decompression layers. Table 5 shows the results. First, we remove the training of the compression independent from the model fine-tuning which causes a significant 6.6 F1 score reduction. Second, we remove the joint training of compression and MRC layers which cause 1.6 F1 score drops. Overall, both stages are necessary for training effective compression layers.

### 5.5 Inference Performance

In addition to FLOPs computational cost analysis, we run inference benchmarks on GPU and CPU. For the benchmarks, we use FP16 PyTorch models without TorchScript. We test in two settings: long and short inputs. Long inputs are 64 words for the question and 448 words for the passage. Short inputs are 16 words for the question and 150 words for the passage. For each setting, we perform four runs and take the average time.
Table 5: Compression ablation study for MRC SQUAD 2.0 model with 5-7 split. We remove either training compression independent of the model or joint training.

Table 6 shows the benchmark results. For CPU the results are close to our FLOPs analysis, and for GPU the we get lower runtime reduction due to the GPU parallelism.

Table 6: Decoupled transformer inference performance. Baseline is a standard transformer MRC model. Long and short indicate the input length, and the times are in milliseconds. Diff is the difference with the baseline.

Table 7: Decoupled transformer with DeBERTa-base and ROaD-base on SQUAD 2.0 model. EM is exact match accuracy. FLOPs is floating point operations for inference as a measure of computational cost.

5.6 Results

Table 7 compares the decoupled ROaD-base 5-7 split model with 4x compression with the ROaD-base model and DeBERTa-base model on SQUAD 2.0 MRC task. The DeBERTa model introduces additional positional embeddings that increase the computational cost by 20%. Still, the decoupled ROaD model achieves comparable accuracy with DeBERTa while requiring two times fewer FLOPs.

6 Conclusion and Future Work

We presented the decoupled transformer model for reducing runtime latency of MRC models in open-domain QA. The decoupling allows for part of the representation computation to be performed offline and cached for online use. To bridge the accuracy gap between a standard transformer and decoupled transformer, we devised knowledge distillation objectives for both model logits and features. Moreover, we introduced a representation compression approach that allows for a four-times reduction in representation storage requirements for open-domain QA without significant loss of accuracy. We use the decoupled transformer to reduce the computational cost of open-domain MRC by 30-40% with only 1.2 points worse than the F1-score on the SQUAD 2.0 benchmark.

In the future, we are planning to extend the decoupled model with a DPR objective. The goal is for the input-component to also produce DPR-like embeddings suitable for similarity search. This way, we can have a single model that acts as both retrieval and reader.

References


Towards Task-Agnostic Privacy-And Utility-Preserving Models

Yaroslav Emelyanov
Sberbank, Moscow, Russia
emelyanov.y.i@sberbank.ru

Abstract
Modern deep learning models for natural language processing rely heavily on large amounts of annotated texts. However, obtaining such texts may be difficult when they contain personal or confidential information, for example, in health or legal domains. In this work, we propose a method of de-identifying free-form text documents by carefully redacting sensitive data in them. We show that our method preserves data utility for text classification, sequence labeling, and question answering tasks.

1 Introduction
Data privacy has become an important topic recently, and new regulations that govern the processing and usage of consumer personal data arise every year. We use the term sensitive data to define all information that contains personal data or other potentially compromising data that can lead to the re-identification of individuals or organizations. With the advent of machine learning, privacy and compliant data governance have become even more important.

Free-form text documents often contain sensitive data. For example, legal documents contain full information about individuals or organizations, dialogues contain references to different people and possibly information that can lead to their identification, such as addresses or job positions. NLP applications such as text classification or question answering require annotated data, and as Feyisetan et al. (2019) argues, the privacy cost of annotating texts must be considered when developing new applications.

Anonymization, or de-identification, is viewed as one of the ways to make working with non-public datasets both safer and more compliant. It is also important to preserve utility in de-identified data for further usages, e.g. machine learning or analytics. Traditional approaches like $k$-anonymity (Sweeney, 2002) were shown to be ineffective when applied to high-dimensional media (Aggarwal, 2005) for example, free-form texts. Moreover, unlike structured data present in relational databases, anonymization of unstructured text documents poses additional challenges, because locations of textual spans which constitute sensitive data must be inferred at runtime.

Previous work either focused on preventing the model from memorizing data (Kerrigan et al., 2020) or considered only the text classification task. In this paper, we aim to solve the challenges of preserving both privacy and utility in free-form text documents with respect to different NLP tasks. We propose a simple “find-and-replace” method for automatically de-identifying documents and evaluate the utility of the de-identified data with respect to different downstream tasks, namely named entity recognition, question answering and text classification. To the best of our knowledge, this is the first work that evaluates the utility of the de-identified data for sequence labeling and reading comprehension tasks.

Specifically, our contributions are as follows:
1. We develop and evaluate the deep-learning-based method for de-identifying text documents.
2. We conduct series of experiments and show that our method mostly preserves utility for different NLP tasks: NER, QA, and text classification.
3. We investigate how our method impacts the end task performance for different model architectures.

2 Background
2.1 Preserving Privacy In Texts
There are several research directions in privacy in NLP. One research direction covers generating synthetic training data, thus abandoning original
data completely. Krishna et al. (2021) shows that this approach performs well for text classification and protects against membership inference attacks. However, in narrow domains such as legal contracts, maintaining internal coherency is important for information retrieval tasks, but generating long coherent texts is still a challenging NLP task (Tan et al., 2021).

Another area of research focuses on adding noise during training to prevent models from memorizing their training data. For example, Kerrigan et al. (2020) add noise to gradients during training to prevent large generative models from "memorizing" training data. However, NLP applications like NER or QA are powered by large amounts of annotated training data and data annotation happens before model training. This means that more people, including annotators, should have access to private data, which adds additional scrutiny to the dataset development process (Feyisetan et al., 2019).

The third branch of privacy research can be viewed as a form of noise added to the original data. The system that adds this noise must satisfy the requirement, stated in Dwork (2006):

Anything that can be learned about a respondent from the statistical database should be learnable without access to the database.

While this assumption is not achievable in practice, because we have to extract some value from the database, several methods were proposed to satisfy more relaxed privacy guarantees, e.g. $(\varepsilon, \delta)$ differential privacy (Geng and Viswanath, 2013).

Finally, a practical approach for texts de-identification via targeting named entities was proposed. In this approach, the auxiliary NER model is trained to recognize sensitive spans in the document that are further redacted. Stubbs et al. (2015); Marimon et al. (2019) organized challenges to develop the best de-identification system for English or Spanish medical texts, however, they did not explore the utility of anonymized data.

Our method falls into the latter research direction, however, we treat text de-identification as an auxiliary part of our work, focusing on measuring utility in the de-identified data.

### 2.2 Measuring Utility In The De-Identified Data

Rahman et al. (2018) show that models trained via privacy-preserving methods may poorly generalize to the original data. This means that data utility is reduced during the de-identification procedure.

Several approaches to measure utility in de-identified data were proposed. For example, Sánchez et al. (2014) follow the information-theoretic approach to ensure that de-identified entities do not exhibit any information that would help the attacker to de-identify data, even when the attacker has access to large amounts of open information from the Internet. Another approach is to measure utility as the quality of models trained on the downstream tasks on the de-identified data (Xu et al., 2020a). While the latter approach gives an intuitive and practical definition of utility, it is not clear how utility estimates depend on the end task, data, and model architecture. We adopt the latter approach and investigate impact of task, data and model choice in the section 4.

### 3 Proposed Method

We aim to develop a method that performs fine-grained substitutions of text spans comprising sensitive information. Contexts of sensitive data are kept intact and therefore our method preserves as much original text as possible.

#### 3.1 Extracting Sensitive Information

Searching for sensitive information, such as names or IDs of individuals, can be difficult in free-form texts. We formulate it as a named entity recognition (NER) task (Mamede et al., 2016), for which various methods were proposed. For example, IDs and other numerical information may be found by regular expressions, while medical diagnoses may be looked up in dictionaries. However, as Yadav and Bethard (2018) suggest, deep learning methods mostly outperform gazetteer-based or feature-based

<table>
<thead>
<tr>
<th>Task</th>
<th>Dataset</th>
<th># instances</th>
<th># classes</th>
<th>domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>NER</td>
<td>RuReBus (Ivanin et al., 2020)</td>
<td>218 documents</td>
<td>8</td>
<td>State documents</td>
</tr>
<tr>
<td>NER</td>
<td>FactRuEval (Starostin et al., 2016)</td>
<td>255 documents</td>
<td>3</td>
<td>News</td>
</tr>
<tr>
<td>Question Answering</td>
<td>SberQuAD (Efimov et al., 2020)</td>
<td>45328 questions</td>
<td>–</td>
<td>Wikipedia</td>
</tr>
<tr>
<td>Text Classification</td>
<td>In-house data</td>
<td>4000 sentences</td>
<td>13</td>
<td>Legal documents</td>
</tr>
</tbody>
</table>

Table 1: Datasets used in experiments
models, and we opt for the neural-network-based approach.

Traditionally (Tjong Kim Sang and De Meulder, 2003), the performance of NER models is evaluated by micro-averaged f1-measure over extracted spans. Several datasets (Stubbs et al. (2015), Garat and Wonsever (2019)) exist to evaluate de-identification models, however, they do not provide annotations that would enable measuring utility in the de-identified data.

### 3.2 Replacing Sensitive Information

Several ways of replacing sensitive information were proposed in the literature (Carrell et al., 2020; Jiang et al., 2019). We study two methods:

1. **Sanitization** strategy redacts the sensitive information and replaces it with the token describing the label of the replaced entity, e.g. replace “John Smith” with generic label PERSON, “Organization inc.” with generic label ORGANIZATION.

2. **Pseudonymization** replaces real entities with synthetically generated but semantically and grammatically sound values. Pseudonymization is often used in practice when releasing private data for research or third parties (Stubbs et al., 2015). Table 2 provides an example of pseudonymization strategy for SberQuAD dataset.

<table>
<thead>
<tr>
<th>Original text</th>
<th>Pseudonymized text</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Context:</strong> В Миссолонги Байрон заболел лихорадкой, продолжая отдавать все свои силы на борьбу за свободу страны.</td>
<td>В Сельниково Колико заболел лихорадкой, продолжая отдавать все свои силы на борьбу за свободу страны.</td>
</tr>
<tr>
<td><strong>Question:</strong> Чем заболел Байрон в Миссолонги?</td>
<td>Чем заболел Колико в Сельниково?</td>
</tr>
<tr>
<td><strong>Answer:</strong> лихорадкой</td>
<td>лихорадкой</td>
</tr>
</tbody>
</table>

Table 2: Consistent pseudonymization for paragraph and question for SberQuad sample and its English translation. All mentions of Байрон (Byron) (highlighted with dashed underline) are pseudonymized consistently. In addition, all mentions of Миссолонги (Missolongi) are also anonymized consistently.

To implement Sanitization strategy, we need only the label of the entity the token corresponds to, which is available from NER model prediction at inference time. However, such replacement erases coreference links throughout the document, which may be important for the downstream task. In section 4 we provide results and show when this strategy impairs the performance on the downstream tasks.

Pseudonymization strategy is more difficult to implement because tokens of different entity types should be replaced differently. For example, a sequence of random digits comprises a number, but sequence of random characters does not always comprise a valid person or organization name. We generate synthetic values for Pseudonymization strategy as follows:

1. For numerical spans, e.g. numbers, IDs, and dates, we generate random numbers of the same length.

2. For textual spans, e.g. names, addresses, we use lookup from dictionaries. We make a random selection from the dictionary independently for every word in the span.

For tasks that require reasoning over input text, like question answering (Rajpurkar et al., 2016), train instances should maintain internal coherency: inconsistent changes in context and question would leave the question unanswerable. To solve this problem, Pseudonymization strategy maintains a mapping between original and pseudonymized values during its work, which allows coherent replacements of entities values. During anonymization of the datasets, each dataset instance is processed independently, meaning that mentions of the same person in different instances will be anonymized differently. We do not add
any coreference information in Sanitization strategy.

### 3.3 Re-Identification Risks

Our de-identification system relies on the NER model. Deleger et al. (2013) show that even double manual de-identification is not perfect, so de-identification errors will inevitably occur. Such errors may lead to re-identification of the de-identified subject, for example, when not all of their mentions were de-identified. Scaiano et al. (2016) propose to use all-or-nothing recall to evaluate de-identification models, because if even one mention of the person within the document was not de-identified, adversary may be able to re-identify the person. For example, if document has 10 mentions of the person, of which 9 were anonymized, then all-or-nothing recall is 0, while regular recall is 0.9. We measure all-or-nothing recall in our experiments. Meystre et al. (2014); Scaiano et al. (2016) argue that re-identification risks are further reduced for Pseudonymization strategy compared to Sanitization: when encountering specific names in the pseudonymized document, it is difficult to tell whether they were left intact by an imperfect anonymizer system or they are the result of pseudonymization.

### 4 Experiments

#### 4.1 Experimental Protocol

Let $S$ be our anonymization strategy (for example, sanitization or pseudonymization) and $D_{target} = (T_{target}, V_{target})$ be the target dataset for which we train model on a downstream task. Target dataset is split into train and validation sets named as $T$ and $V$. Let $M$ be the model trained on $T_{target}$. Our goal is to evaluate how $M$’s performance on $V_{target}$ depends on the dataset $M$ is trained on: original $T_{target}$ or de-identified $T’_{target}$. Inspired by this goal, we design the following experimental protocol:

1. Anonymize $T_{target}$, $V_{target}$ and obtain $T’_{target} = S(T_{target})$, $V’_{target} = S(V_{target})$
2. Train model on original data: $M = M(T_{target})$, get validation metrics $M(V_{target})$
3. Train model on de-identified data: $M’ = M(T’_{target})$, get validation metrics for both original and de-identified data: $M’(V’_{target})$, $M’(V_{target})$
4. Compare results of models: $\Delta = M’(V_{target}) - M(V_{target})$

We repeat this experiment for every strategy $S$ and compare results for each strategy with the baseline that was trained on the original data. Our anonymization procedure may yield distribution shift which may result in imperfect generalization to the original validation set and therefore in negative values of $\Delta$. $\Delta \geq 0$ means that data utility was completely preserved. We repeat each experiment 3 times and report mean and variance for $M’(V_{target})$ across them in Table 3.1.

#### 4.2 NER Models For Extracting Sensitive Data

For experiments on publicly available data, we use Collection3 corpus created by Mozharova and Loukachevitch (2016) to develop our anonymizer system. This corpus has the same annotation schema as FactRuEval and is approximately 7 times larger. We train vanilla BERT-based NER
model on Collection3 corpus and obtain the micro f1 measure 0.931. To measure re-identification risks, we manually review 200 randomly chosen documents from anonymized SberQuAD train set and find that all-or-nothing recall is 0.93.

For NER model for in-house data de-identification, we use an in-house corpus of 3040 documents with 426 272 annotated entities. This corpus has annotation schema similar to Ontonotes dataset (Weischedel et al., 2011). We train BERT-CRF model on this corpus using 90-10 train-test split, evaluate the model using micro f1-measure over spans and get the value of 0.93. We performed an additional evaluation to see how well our model finds sensitive data. We asked domain experts to manually annotate sensitive information in 30 legal documents of various types and then checked our system’s output against these annotations. We found that of 1030 entities, 1009 were anonymized, resulting in 0.98 recall and 0.95 all-or-nothing recall.

4.3 Downstream Tasks And Models

To show that our method transfers across tasks and domains, we use different NLP tasks and datasets:

**FactRuEval** (Starostin et al., 2016) is a NER dataset developed to evaluate fact extraction from Russian news articles. It is annotated similarly to Tjong Kim Sang and De Meulder (2003) and has PER, LOC, ORG entities.

**RuReBus** (Ivanin et al., 2020) is another NER dataset consisting of state documents and reports. It has more diverse annotation schema then FactRuEval. It is annotated with custom annotation schema that includes entities like METRICS, ACTIVITY, QUALITATIVE. Unlike FactRuEval, most of the classes in this dataset are not considered as sensitive data, except for INSTITUTION class, which is similar to ORG class in FactRuEval.

**SberQuAD** (Efimov et al., 2020) is Russian extractive question answering dataset similar to SQuAD (Rajpurkar et al., 2016). It has 9,080 unique paragraphs and 50,364 questions, about 20% of answers and about 72% of paragraphs contain named entities that should be anonymized, e.g., people, locations, or organizations. Unlike Rajpurkar et al. (2016), SberQuAD does not have unanswerable questions.

For the text classification task, an internal dataset of 5,000 texts annotated with 13 different classes was used. Data instances are segments of legal documents and classes represent types of these segments.

We use train-dev splits provided by the authors for all publicly available datasets. For text classification and NER tasks, we use micro averaged f1 measure. For SberQuAD, we use f1 measure as in the SQuAD dataset.

Comparative statistics of all datasets are shown in Table 1.

For NER datasets, we use simple BERT for token classification as described in Devlin et al. (2019). For SberQuAD, we use the same architecture as Devlin et al. (2019) used for SQuAD. For the in-house text classification dataset, BERT performed on par with gradient boosting (Ke et al., 2017) on top of tf-idf vectorization, and we choose boosting for its simplicity. It is also interesting to investigate how our anonymization methods affect end task performance for different vectorization methods.

4.4 Utility Tests

As described in subsection 4.1, we measure the utility of anonymized data as the performance drop between models trained on original and anonymized datasets. All experiments were implemented with AllenNLP (Gardner et al., 2018) framework. We present our results in Table 3.1.

In all experiments, baseline models trained on original data achieved performance close to currently reported state-of-the-art results. We note that in all experiments anonymization impairs end task performance, although results vary depending on the task and dataset.

Experiments on RuReBus dataset showed only a slight performance drop, however, all models achieve relatively low scores compared to the FactRuEval dataset. We attribute low scores to the inconsistent annotations in the RuReBus dataset. We attribute the low difference between performance on pseudonymized and sanitized data to the annotation schema: most entities in the schema are not considered sensitive information. However, for entity INSTITUTION, which is close to ORG entity, performance drop is significant: from 0.436 f1-measure in original data to 0.348 in sanitized data.

Similarly, in text classification task performance changes are also small compared to SberQUAD and FactRuEval tasks. This can be explained by the nature of the task: Xu et al. (2020b); Marivate and Sefara (2020) show that text classi-
**Table 4:** Results for different architectures for FactRuEval dataset. Higher Δ is better.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Training data</th>
<th>M'(T_{tgt})</th>
<th>M'(V_{tgt})</th>
<th>M'(V_{tgt})</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-FC</td>
<td>original</td>
<td>0.99</td>
<td>0.85</td>
<td>0.85</td>
<td>-</td>
</tr>
<tr>
<td>BERT-FC</td>
<td>sanitized</td>
<td>0.98</td>
<td>0.82</td>
<td>0.40</td>
<td>-0.45</td>
</tr>
<tr>
<td>BERT-FC</td>
<td>pseudonimized</td>
<td>0.99</td>
<td>0.81</td>
<td>0.76</td>
<td>-0.09</td>
</tr>
<tr>
<td>BERT-BiLSTM</td>
<td>original</td>
<td>0.96</td>
<td>0.75</td>
<td>0.77</td>
<td>-</td>
</tr>
<tr>
<td>BERT-BiLSTM</td>
<td>sanitized</td>
<td>0.93</td>
<td>0.77</td>
<td>0.15</td>
<td>-0.62</td>
</tr>
<tr>
<td>BERT-BiLSTM</td>
<td>pseudonimized</td>
<td>0.95</td>
<td>0.70</td>
<td>0.61</td>
<td>-0.16</td>
</tr>
<tr>
<td>w2v-CNN-BiLSTM</td>
<td>original</td>
<td>0.89</td>
<td>0.70</td>
<td>0.69</td>
<td>-</td>
</tr>
<tr>
<td>w2v-CNN-BiLSTM</td>
<td>sanitized</td>
<td>0.83</td>
<td>0.72</td>
<td>0.12</td>
<td>-0.57</td>
</tr>
<tr>
<td>w2v-CNN-BiLSTM</td>
<td>pseudonimized</td>
<td>0.77</td>
<td>0.55</td>
<td>0.46</td>
<td>-0.23</td>
</tr>
</tbody>
</table>

The factification task is robust to different kinds of noise, including word substitution, which is close to our anonymization procedure.

Results on SberQuAD dataset confirm our hypothesis that consistent anonymization is important for question answering: difference between pseudonymized and sanitized data is higher than in previous experiments. About 28% of the dataset was kept intact by anonymization and therefore performance drop for anonymized instances will be even larger.

The largest difference in performance between models trained on sanitized and pseudonymized data is in FactRuEval dataset. This difference can be attributed to the annotation schema and nature of the task: Bernier-Colborne and Langlais (2020) showed that NER models rely more on entity text and less on the entity context. This intuition explains performance drop for models trained on sanitized data: they have not seen any real entities during training and can find entities based only on the context during the evaluation on the original data. However, models trained on pseudonymized data are able to generalize from synthetic entities to real ones.

Our experiments suggest that task, data and annotation schema impact downstream task model sensitivity to data anonymization.

### 4.5 Impact Of The Downstream Model Architecture

In lieu of the current NLP state, we perform most of our experiments using BERT-based models. However, we also explore how anonymization impacts end task performance for different model architectures. We choose FactRuEval dataset for this experiment because in prior experiments we showed that it is more sensitive to data anonymization. We use three popular NER architectures:

**BERT-FC** is a vanilla BERT for token classification model (Devlin et al., 2019), both pre-trained layers and projection layer are fine-tuned during training. We use *RuBERT* initialization trained by Kuratov and Arkhipov (2019) in all experiments because it performed the best on the original data in all experiments.

**BERT-BiLSTM** is an architecture with the 2-layer bidirectional LSTM applied on top of BERT embeddings. During training, BERT parameters are frozen and only LSTM layers are tuned. We use the same *RuBERT* initialization.

**w2v-CNN-BiLSTM** is a popular architecture that uses fixed word embeddings together with character embeddings to encode each token and BiLSTM on top of them to encode context. We use word embeddings trained by Grave et al. (2018) and keep them frozen during training due to the small size of the training corpus.

We provide results in Table 4. As in subsection 4.4, pseudonymization enjoys lower performance drops for all architectures. We notice that performance drops for BERT-based models are lower. This can be explained by the number of OOV words that generates our pseudonymization procedure: synthetic names or addresses are randomly sampled from the large dictionaries, so they are mostly not present in the embeddings table even for large pre-trained word embeddings. We calculated that only 38% of all names and 15% of all surnames from our dictionaries are present in the pre-trained embeddings. Our results support the claim made by Hendrycks et al. (2020), who showed that pre-trained transformers are more robust to distribution shifts.

### 5 Suggestions To Practitioners

Our experiments highlight several characteristics of anonymization procedure, downstream task and
model architecture that should be taken into account when using anonymized data for training NLP models. Our suggestions are as follows:

1. Noise-robust downstream tasks are also robust to anonymization.

2. Downstream tasks that do not require reasoning over named entities are robust to anonymization.

3. Downstream tasks that require reasoning over named entities also require coherent pseudonymization to maintain data consistency.

4. Pseudonymization works better than sanitization, although it is more difficult to develop.

5. Transformer-based models generalize the best between original and anonymized data.

6 Conclusions And Future Work

In this work, we consider the practical side of anonymizing unstructured documents while simultaneously preserving their utility for different downstream tasks. New policies regarding personal data make privacy research a more important topic over the years. We anticipate that in the near future de-identification of sensitive data before training will become a necessity. We hope our work will pave the way for investigating broader impact and limitations of free-form text anonymization.

We demonstrate that pseudonymization mostly preserves data utility for different extractive NLP tasks. We show it is possible to achieve close results with the model trained only on the de-identified data. However, it is not yet clear whether our results transfer to generative tasks and more complex settings, for example, scenarios with multiple languages like machine translation or multilingual datasets. We believe this is the promising research direction.

References


Quan Geng and P. Viswanath. 2013. The optimal mechanism in $(\epsilon, \delta)$-differential privacy.


Valerie Mozharova and Natalia Loukachevitch. 2016. Two-stage approach in russian named entity recognition. Institute of Electrical and Electronics Engineers Inc.


Knowledge Discovery in COVID-19 Research Literature

Ernesto L. Estevanell-Valladares¹, Alejandro Piad-Morffis¹, Suilan Estevez-Velarde¹, Yoan Gutierrez², Andres Montoyo², Rafael Muñoz², and Yudivian Almeida-Cruz¹

¹School of Math and Computer Science, University of Havana
{estevanell,apiad,sestevez,yudivian}@matcom.uh.cu
²Department of Languages and Computing Systems, University of Alicante
{ygutierrez,montoyo,rafael}@dlsi.ua.es

Abstract

This paper presents the preliminary results of an ongoing project that analyzes the growing body of scientific research published around the COVID-19 pandemic. In this research, a general-purpose semantic model is used to double annotate a batch of 500 sentences that were manually selected from the CORD-19 corpus. Afterwards, a baseline text-mining pipeline is designed and evaluated via a large batch of 100,959 sentences. We present a qualitative analysis of the most interesting facts automatically extracted and highlight possible future lines of development. The preliminary results show that general-purpose semantic models are a useful tool for discovering fine-grained knowledge in large corpora of scientific documents.

1 Introduction

The COVID-19 pandemic that affected almost every country at the beginning of 2020 has provoked a massive increase in scientific papers related to biomedical sciences (Velavan and Meyer, 2020). The scientific community’s effort to fight the spread of the virus is evident in the record number of research papers about COVID-19 that have been submitted to conferences, journals and preprint services. Several academic publishers joined the effort by providing free access to research in related areas that could be useful to scientists and academics. The amount of information produced during this period greatly surpassed the ability of human researchers to stay up-to-date, which in turn spawned an increased interest in the application of computational techniques to automatically organize, normalize and link the existing information.

A recent initiative is the COVID-19 Open Research Dataset (CORD-19), published by the Allen Institute for AI (Lo et al., 2020), which makes available a large corpus of scientific papers on COVID-19 and related topics. At the moment of writing, it contains 76,674 scientific articles. The corpus has been used as part of a Kaggle challenge¹, which focused mainly on unsupervised tasks related to organizing and categorizing the different aspects of the whole COVID-19 situation. Based on these resources, computational tools, e.g., SciSight (Hope et al., 2020), SciFact (Wadden et al., 2020), and similar (Bras et al., 2020), have been created to enable the interactive visualization and exploration of the scientific literature and the discovery of connections between the available methods, symptoms, interventions, etc.

Unsupervised approaches are a natural strategy for dealing with large, unlabeled corpora, while supervised approaches have the caveat of requiring training examples to be manually annotated, but can provide precise answers to specific questions given enough supervised data. For example, identifying domain-specific entities such as symptoms, medication and treatments, and semantic relations between them. Learning to recognize this type of information in natural language, even academic language, is a challenging task, given the large number of varieties in which the same semantic fact can be stated. In this context, different annotation models have been designed to capture the semantic meaning in different domains and levels of discourse. Token-level annotation models, such as AMR (Banarescu et al., 2013), capture fine-grained semantic relations between elements in a natural language sentence, independently of domain, which means that only general-purpose relations can be recognized. In contrast, domain-specific annotation models can capture more detailed relations, such as .
Phylogenetic studies have shown that 2019-nCoV and SARS-CoV belong to the subgenus Sarbecovirus, but they are distantly related, with a sequence identity of 79.6% at the whole-genome level. However, SARS-CoV, using the same receptor, was not detected in skeletal muscle by post-mortem examination.

In conclusion, SARS-CoV is the closest related virus to 2019-nCoV for which a significant number of epitopes has been defined in humans (and other species), and that also causes human disease with lethal outcomes.

The main objective of this research is to build a manually annotated corpus that can be used to train knowledge discovery systems for analyzing the COVID-19 research. For this purpose, we apply the SAT+R annotation model to English sentences in the CORD-19 corpus, manually annotating a small training set that can bootstrap a text-mining process for the entire corpus. The main contributions of this research are:

- The manual annotation of 500 sentences hand-picked from the CORD-19 corpus with the SAT+R annotation model, using a volunteer crowd-based approach with 2 versions of each annotation made by different non-expert annotators.
- The implementation and evaluation of a baseline text-mining pipeline trained on the manually annotated sentences.
- The application of the text-mining pipeline to the full CORD-19 corpus with an analysis of the most relevant concepts and relations discovered.
- All the relevant data to replicate and continue this research, including the source code and annotated corpus in BRAT Standoff format (Stenetorp et al., 2012), are available online for the research community.

The remainder of this paper is organized as follows. Section 2 introduces the SAT+R annotation model and presents the manually annotated corpus, its statistics and quality metrics, and details about the annotation process. Section 3 presents and evaluates a baseline machine learning pipeline to automatically annotate the raw text in the CORD-19 corpus. Section 4 describes the most relevant concepts and relations automatically extracted from the a larger fraction of CORD-19 corpus. Finally, Section 5 discusses the findings and lessons learned during this research and highlights possible lines of development, and Section 6 presents the conclusions.

2 Corpus Description

This section introduces the CORD-ANN corpus, an ongoing linguistic resource which contains semantically annotated sentences extracted from COVID-related research papers in the CORD-19 corpus. The corpus is constructed following the SAT+R annotation model (Piad-Morffis et al., 2019a), which is based around Concepts linked by Actions, and additional ontological and teleological relations, such as is-a, part-of, etc.
<table>
<thead>
<tr>
<th>Metric</th>
<th>Total</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentences</td>
<td>1000</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Entities</td>
<td>10201</td>
<td>5110</td>
<td>5091</td>
</tr>
<tr>
<td>Concept</td>
<td>8231</td>
<td>4154</td>
<td>4077</td>
</tr>
<tr>
<td>Action</td>
<td>1868</td>
<td>916</td>
<td>952</td>
</tr>
<tr>
<td>Reference</td>
<td>102</td>
<td>40</td>
<td>62</td>
</tr>
<tr>
<td>Relations</td>
<td>9444</td>
<td>4735</td>
<td>4709</td>
</tr>
<tr>
<td>in-context</td>
<td>2239</td>
<td>1077</td>
<td>1162</td>
</tr>
<tr>
<td>has-property</td>
<td>2185</td>
<td>1146</td>
<td>1039</td>
</tr>
<tr>
<td>target</td>
<td>1686</td>
<td>845</td>
<td>841</td>
</tr>
<tr>
<td>subject</td>
<td>1269</td>
<td>640</td>
<td>629</td>
</tr>
<tr>
<td>is-a</td>
<td>647</td>
<td>322</td>
<td>325</td>
</tr>
<tr>
<td>in-place</td>
<td>366</td>
<td>177</td>
<td>189</td>
</tr>
<tr>
<td>causes</td>
<td>297</td>
<td>161</td>
<td>136</td>
</tr>
<tr>
<td>has-part</td>
<td>217</td>
<td>114</td>
<td>103</td>
</tr>
<tr>
<td>entails</td>
<td>207</td>
<td>88</td>
<td>119</td>
</tr>
<tr>
<td>in-time</td>
<td>170</td>
<td>87</td>
<td>83</td>
</tr>
<tr>
<td>same-as</td>
<td>161</td>
<td>78</td>
<td>83</td>
</tr>
</tbody>
</table>

| Attributes | 606   | 329 | 277 |
|           | 228   | 119 | 109 |
|           | 204   | 114 | 90  |
|           | 142   | 78  | 64  |
|           | 32    | 18  | 14  |

Table 1: Summary statistics for the CORD-ANN corpus.

as is-a, has-property, causes, entails, among others. During an initial annotation trial it was identified that Predicates, one of the semantic types defined in SAT+R, produced a large degree of disagreement among annotators while providing little additional information, and for this reason this element is not considered. Figure 1 shows three real sentences annotated with a variety of the semantic elements defined in the SAT+R annotation model.

Table 1 shows the total number of annotated elements. The annotated corpus has 500 sentences manually selected from the CORD-19 corpus. Each sentence was manually annotated by 2 different annotators who where not allowed to share their annotations. Table 1 shows a fine-grained description of the corpus annotations. A total of 10, 201 entities and 9, 444 relations were annotated, averaging 10.20 entities and 9.44 relations per sentence.

A manual pre-selection process was carried out to choose the sentences that contain the most relevant content. The annotation process is realized in batches of 5 sentences, given the linguistic complexity of the academic language of the corpus and the fact that the annotators are not native English speakers. That being said, the majority of the annotators were undergraduate student or graduate students of different university degrees with at least a B2 English level. The annotation procedure was adapted from the methodology proposed by Piad-Morffis et al. (2019a). The tool used for annotation is BRAT (Stenetorp et al., 2012) given the simplicity of its user interface. Configuration files and related infrastructure are published in the project repository.

The annotators were recruited through social media and most are from the academic institutions to which the authors are affiliated. A total of 21 different annotators were involved in the corpus creation, although several more showed some degree of interest but didn’t complete any annotation batch. The degree of involvement varied widely, since two annotators account for approximately half of the corpus (51 and 48 batches respectively) while 9 annotators submitted only one batch. An annotation guide with several examples was published online, and the first batch from every annotator was cross-checked by the authors to provide feedback. Afterwards, a continuous annotation campaign was managed through social media, with regular periods in which the annotators joined in an online forum to ask for clarifications or share their suggestions.

The annotation process was carried out from March 28th until June 9th, when the first 500 sentences were completed. At the moment of writing, the annotation campaign has been temporarily halted in order to analyze the partial results obtained and decide the best course of action for the continuity of this research.

Table 2 shows the agreement score between each pair of annotations for each type of semantic element. The metrics reported are precision, recall and $F_1$ computed as a micro-average between every pair of sentences doubly annotated. Since the $F_1$ metric is symmetric with respect to precision and recall, these are taken with respect to an arbitrary first annotator for each sentence. Overall, the agreement for entities is higher than for relations. The most difficult semantic relations to annotate, in terms of agreement, are entails, causes and has-part, while the easiest are the teleological relations subject and target, followed by the ontological relations is-a and has-property.
3 Baseline Text-Mining Pipeline

This section presents a simple machine learning pipeline for the automatic annotation of entities and relations in raw sentences from the CORD-19 corpus following the annotation model described in Section 2. This pipeline is trained on the 1,000 manually annotated sentences (i.e., the two versions of each annotated sentence), and executed on the remaining of the CORD-19 corpus. A high-level overview of the pipeline, shown in Figure 2, is composed of the following steps:

1. Sentences are tokenized and syntactic and morphological features are extracted from each token (using the spaCy\textsuperscript{4} library).

2. The annotated entities are converted from BRAT’s Standoff format to a BILOUV encoding (i.e., \textit{Begin}, \textit{Inside}, \textit{Last}, \textit{Out}, \textit{Unit} and \textit{Overlap}).

3. A CRF model $M_E$ is trained on the token features to predict the BILOUV encoding.

4. Each relation pair is converted to a set of aggregated features, and negative relation pairs are randomly sampled.

5. A linear model (logistic regression) $M_R$ is trained on relation pairs to predict the 10 Relation classes in Table 1 plus and additional NONE relation label.

6. The entity model $M_E$ is executed on unlabeled sentences and the result is converted from BILOUV encoding to BRAT’s Standoff format.

7. The relation model $M_R$ is executed on the pairs of entities predicted in the previous step.

For the entity model $M_E$, the syntactic and morphological features include lemma, coarse and fine-grained part-of-speech, dependency labels, general-purpose entity labels (e.g., PERSON, LOCATION, etc.), word shape, and several flags for specific patterns such as emails, numbers, and URLs. For the relation model $M_R$, the aggregated features correspond to the features of the tokens that comprise the two entities that participate in the relation, as well as the features of all the tokens in the smallest sub-tree of the dependency tree that contains both entities.

The ultimate purpose of these models is to automatically extract relevant knowledge from the unlabeled pool of sentences. Taking into account the complexity of this natural language comprehension task, there is always a trade-off between extracting as much knowledge as possible (i.e., maximizing recall) versus extracting knowledge as accurately as possible (i.e., maximizing precision). However, this trade-off can be explicitly controlled by measuring a degree of uncertainty $\sigma$ in the models’ predictions, and only outputting the elements (i.e., entities and relations) whose uncertainty is below a given threshold $\sigma^*$. For the entity model $M_E$, the raw marginal probabilities provided by the CRF

\begin{table}[h]
\centering
\begin{tabular}{|l|c|c|c|}
\hline
Annotation & Recall & Precision & $F_1$ \\
\hline
Entities & 0.6367 & 0.6418 & 0.6392 \\
Concept & 0.6699 & 0.6852 & 0.6775 \\
Action & 0.5027 & 0.5424 & 0.5218 \\
Reference & 0.2625 & 0.7000 & 0.3818 \\
\hline
Relations & 0.4875 & 0.4982 & 0.4928 \\
target & 0.6416 & 0.6381 & 0.6398 \\
subject & 0.6094 & 0.6418 & 0.6252 \\
has-property & 0.5454 & 0.6085 & 0.5752 \\
is-a & 0.5333 & 0.5303 & 0.5318 \\
in-context & 0.4119 & 0.3806 & 0.3956 \\
in-time & 0.3333 & 0.3513 & 0.3421 \\
causes & 0.1264 & 0.1692 & 0.1447 \\
has-part & 0.1142 & 0.1538 & 0.1311 \\
entails & 0.0909 & 0.0789 & 0.0845 \\
\hline
\end{tabular}
\caption{Relative agreement between annotators for each type of annotation.}
\end{table}

\textsuperscript{4}https://spacy.io
Table 3: Results of 30 independent evaluations for the entity model $M_E$ and relation model $M_R$ aggregated for different values of the uncertainty threshold $\sigma^*$. The table includes the precision ($\text{Prec.}$) and recall ($\text{Rec.}$) values for each threshold, as well as the F1 score ($F_1$).

<table>
<thead>
<tr>
<th>Threshold ($\sigma^*$)</th>
<th>$M_E$ Prec.</th>
<th>$M_E$ Rec.</th>
<th>$M_R$ Prec.</th>
<th>$M_R$ Rec.</th>
<th>$F_1$</th>
<th>$M_R$ Prec.</th>
<th>$M_R$ Rec.</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.000</td>
<td>0.200</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.50</td>
<td>0.013</td>
<td>0.887</td>
<td>0.025</td>
<td>0.000</td>
<td>0.000</td>
<td>0.013</td>
<td>0.887</td>
<td>0.025</td>
</tr>
<tr>
<td>0.75</td>
<td>0.048</td>
<td>0.718</td>
<td>0.090</td>
<td>0.082</td>
<td>0.267</td>
<td>0.048</td>
<td>0.718</td>
<td>0.090</td>
</tr>
<tr>
<td>1.00</td>
<td>0.086</td>
<td>0.585</td>
<td>0.150</td>
<td>0.138</td>
<td>0.695</td>
<td>0.086</td>
<td>0.585</td>
<td>0.150</td>
</tr>
<tr>
<td>1.25</td>
<td>0.134</td>
<td>0.539</td>
<td>0.214</td>
<td>0.231</td>
<td>0.457</td>
<td>0.134</td>
<td>0.539</td>
<td>0.214</td>
</tr>
<tr>
<td>1.50</td>
<td>0.196</td>
<td>0.544</td>
<td>0.289</td>
<td>0.227</td>
<td>0.326</td>
<td>0.196</td>
<td>0.544</td>
<td>0.289</td>
</tr>
<tr>
<td>1.75</td>
<td>0.300</td>
<td>0.552</td>
<td>0.388</td>
<td>0.232</td>
<td>0.270</td>
<td>0.300</td>
<td>0.552</td>
<td>0.388</td>
</tr>
<tr>
<td>2.00</td>
<td>0.437</td>
<td>0.543</td>
<td>0.484</td>
<td>0.244</td>
<td>0.243</td>
<td>0.437</td>
<td>0.543</td>
<td>0.484</td>
</tr>
<tr>
<td>2.25</td>
<td>0.541</td>
<td>0.519</td>
<td>0.530</td>
<td>0.241</td>
<td>0.223</td>
<td>0.541</td>
<td>0.519</td>
<td>0.530</td>
</tr>
<tr>
<td>2.50</td>
<td>0.585</td>
<td>0.506</td>
<td>0.543</td>
<td>0.238</td>
<td>0.212</td>
<td>0.585</td>
<td>0.506</td>
<td>0.543</td>
</tr>
<tr>
<td>2.75</td>
<td>0.592</td>
<td>0.502</td>
<td>0.543</td>
<td>0.237</td>
<td>0.210</td>
<td>0.592</td>
<td>0.502</td>
<td>0.543</td>
</tr>
</tbody>
</table>

Table 4: Maximum precision, recall and $F_1$ obtained for different annotation elements. Each value was obtained for a potentially different uncertainty threshold.

<table>
<thead>
<tr>
<th>Annotation</th>
<th>$\text{Prec.}$</th>
<th>$\text{Rec.}$</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>0.961</td>
<td>0.221</td>
<td>0.327</td>
</tr>
<tr>
<td>Concept</td>
<td>0.887</td>
<td>0.684</td>
<td>0.576</td>
</tr>
<tr>
<td>Reference</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>causes</td>
<td>0.133</td>
<td>0.092</td>
<td>0.104</td>
</tr>
<tr>
<td>entails</td>
<td>0.008</td>
<td>0.005</td>
<td>0.006</td>
</tr>
<tr>
<td>has-part</td>
<td>0.112</td>
<td>0.040</td>
<td>0.052</td>
</tr>
<tr>
<td>has-property</td>
<td>0.572</td>
<td>0.282</td>
<td>0.269</td>
</tr>
<tr>
<td>in-context</td>
<td>0.716</td>
<td>0.352</td>
<td>0.399</td>
</tr>
<tr>
<td>in-place</td>
<td>0.320</td>
<td>0.131</td>
<td>0.171</td>
</tr>
<tr>
<td>in-time</td>
<td>0.254</td>
<td>0.077</td>
<td>0.105</td>
</tr>
<tr>
<td>is-a</td>
<td>0.371</td>
<td>0.282</td>
<td>0.296</td>
</tr>
<tr>
<td>subject</td>
<td>0.268</td>
<td>0.373</td>
<td>0.305</td>
</tr>
<tr>
<td>target</td>
<td>0.330</td>
<td>0.460</td>
<td>0.366</td>
</tr>
</tbody>
</table>

Figure 3: Maximum precision achieved for different semantic elements given a specific uncertainty threshold. Only results with recall above 0.1 are considered.

To better understand the trade-off between precision and recall, Figure 3 shows the precision obtained at different uncertainty thresholds for each annotation element. However, since a very high precision can be achieved with an arbitrarily low recall, we only consider annotations for which the average recall is above 0.1 at the given threshold level. This guarantees that at least a 10% of the potential number of those semantic elements would be extracted from the unlabeled collection.

4 Preliminary Insights in Knowledge Discovery

This section presents a qualitative analysis of a knowledge discovery process that can be performed using machine learning models trained on this type of annotated data. For this purpose, the remaining sentences of the CORD-19 corpus that were not used during the annotation process were fed to the machine learning models $M_E$ and $M_R$ and all pre-
Table 5: Total number of instances extracted from the unlabeled sentences.

<table>
<thead>
<tr>
<th>Type</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentences</td>
<td>100,959</td>
</tr>
<tr>
<td>Entities</td>
<td>782,141</td>
</tr>
<tr>
<td>Concept</td>
<td>737,838</td>
</tr>
<tr>
<td>Action</td>
<td>44,289</td>
</tr>
<tr>
<td>Reference</td>
<td>14</td>
</tr>
<tr>
<td>Relations</td>
<td>783,534</td>
</tr>
<tr>
<td>has-property</td>
<td>360,986</td>
</tr>
<tr>
<td>in-context</td>
<td>267,333</td>
</tr>
<tr>
<td>target</td>
<td>72,730</td>
</tr>
<tr>
<td>is-a</td>
<td>26,070</td>
</tr>
<tr>
<td>subject</td>
<td>22,513</td>
</tr>
<tr>
<td>same-as</td>
<td>15,520</td>
</tr>
<tr>
<td>in-place</td>
<td>12,824</td>
</tr>
<tr>
<td>entails</td>
<td>2,058</td>
</tr>
<tr>
<td>causes</td>
<td>1,805</td>
</tr>
<tr>
<td>in-time</td>
<td>895</td>
</tr>
<tr>
<td>has-part</td>
<td>800</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Concept</th>
<th>Instances</th>
<th>Concept</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>cases</td>
<td>5,742</td>
<td>COVID-19</td>
<td>3,007</td>
</tr>
<tr>
<td>number</td>
<td>4,775</td>
<td>infection</td>
<td>3,006</td>
</tr>
<tr>
<td>patients</td>
<td>4,577</td>
<td>cell</td>
<td>2,411</td>
</tr>
<tr>
<td>SARS</td>
<td>4,284</td>
<td>epidemic</td>
<td>2,315</td>
</tr>
<tr>
<td>model</td>
<td>3,755</td>
<td>proteins</td>
<td>2,224</td>
</tr>
<tr>
<td>data</td>
<td>3,686</td>
<td>human</td>
<td>2,223</td>
</tr>
<tr>
<td>protein</td>
<td>3,382</td>
<td>RNA</td>
<td>2,169</td>
</tr>
<tr>
<td>virus</td>
<td>3,222</td>
<td>CoV</td>
<td>2,168</td>
</tr>
<tr>
<td>granted</td>
<td>3,147</td>
<td>infected</td>
<td>2,153</td>
</tr>
<tr>
<td>cells</td>
<td>3,015</td>
<td>China</td>
<td>2,132</td>
</tr>
</tbody>
</table>

Table 6: Most common entities extracted from the CORD-19 corpus.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Source</th>
<th>Destination</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>is-a</td>
<td>SARS</td>
<td>coronavirus</td>
<td>85</td>
</tr>
<tr>
<td>is-a</td>
<td>MERS</td>
<td>CoV</td>
<td>76</td>
</tr>
<tr>
<td>is-a</td>
<td>SEIR</td>
<td>model</td>
<td>64</td>
</tr>
<tr>
<td>is-a</td>
<td>influenza</td>
<td>virus</td>
<td>61</td>
</tr>
<tr>
<td>is-a</td>
<td>A549</td>
<td>cells</td>
<td>46</td>
</tr>
<tr>
<td>has-property</td>
<td>cases</td>
<td>confirmed</td>
<td>439</td>
</tr>
<tr>
<td>has-property</td>
<td>patients</td>
<td>severe</td>
<td>357</td>
</tr>
<tr>
<td>has-property</td>
<td>cases</td>
<td>severe</td>
<td>273</td>
</tr>
<tr>
<td>has-property</td>
<td>number</td>
<td>basic</td>
<td>245</td>
</tr>
<tr>
<td>has-property</td>
<td>cases</td>
<td>imported</td>
<td>213</td>
</tr>
</tbody>
</table>

Table 7: Most common instances for the relations is-a and has-property.

dicted entities and relations were stored. Table 5 presents the total number of sentences processed as well as entities and relations extracted. Given the relatively low performance of the machine learning models, a simple post-processing was introduced to remove the most obvious sources of errors, such as numbers and mathematical symbols, that were incorrectly detected as entities. As expected, the distribution of extracted elements closely follows the distribution of annotations in the training set (see Table 1).

The 20 most common entities extracted are summarized in Table 6. Unsurprisingly, they correspond to common concepts in the medical literature related to epidemics, treatments, biological entities, as well as some COVID-specific entities and locations. Similarly, Table 7 shows the most common instances of the relations is-a and has-property, which are the most basic ontological relations. As expected, they correspond mostly to known relations in the medical domain and specifically in the COVID-related literature.

Finally, Figure 4 shows a cherry-picked graph of relations built around the concept COVID-19. This graph was constructed by sampling the most common relations that involve this concept, manually eliminating irrelevant tuples, such as in-context, and compacting very similar relations (in terms of lemma) into the same nodes. The graph shows interesting relations, such as known symptoms (grouped under the causes relation) and a number of properties that are reported among the biomedical literature in the CORD-19 corpus.

5 Discussion and Future Work

This section discusses two important insights that arise from this ongoing research. First, we analyze the quantity and quality of the extracted knowledge, in an attempt to validate the approach and estimate the impact of its components. Second, we discuss some lessons learned during the annotation process in the hope that future research can further improve on our work.

Arguably, two of the most relevant annotation patterns for the purpose of knowledge discovery are ontological relations such as is-a and has-property, and the Subject-Action-Target triplets via the target and subject relations. The fact that these 4 relations have a relatively high number of instances extracted is promising. In contrast, the relations en-
tails and causes show a significantly lower number of instances, as well as a lower performance. These relations are important from the point of view of knowledge discovery since they could directly link symptoms and treatments with evidences. However, more work is necessary to achieve a reasonable level of performance in their extraction.

The machine learning models presented in this work were designed as an initial baseline. In a practical scenario more powerful models would be deployed. Possible strategies include using a separate CRF model for each entity class, and using a hierarchical model for the relation extraction step. Another possible strategy is to repurpose existing models deployed in similar tasks. According to the 2020 edition of the eHealth-KD Challenge, the top performing deep learning models in this task achieve $F_1$ values of 0.82 and 0.63 in entity and relation extraction, respectively (Piad-Morffis et al., 2020). Since they are also built on the SAT+R annotation model, their adaptation to the corpus used in this research is straightforward. However, it should be considered that the reduced number of training examples in our corpus presents a significant challenge for any machine learning model, which motivates the use of transfer learning techniques. In this sense, additional annotated corpora with SAT+R, such as the one presented in Piad-Morffis et al. (2020) can be used to bootstrap such a system.

Moving forward towards a fully-fledged ontology learning process still requires a significant effort after the annotation and model training. Particularly, since each entity can be potentially represented in the text in different forms, a normalization step would be necessary to group all mentions of similar entities under a single concept. At the moment of writing we are developing approaches for automatic normalization of entity mentions based on Wikidata, but the results are still not available. Furthermore, in this work we have not taken into account the prediction confidence beyond its use as a threshold. Weighting the number of mentions of each entity and relation with their respective
confidence could provide an additional metric to determine what to include in the knowledge graph.

One surprising and positive conclusion of our research is that crowd-based annotation efforts like this one, even with complex cognitive tasks involving deep semantics, are feasible. The annotators that participated voluntarily in this research were motivated by purely altruistic reasons, since there was no monetary incentive. We argue that, even if the COVID-19 pandemic played a large role in this motivation, in general people can be motivated for this kind of work because of the positive social impact of the research. Regarding annotation quality, there is a non-trivial amount of initial tutoring and feedback necessary, but once an annotator acquires a certain level of expertise, these efforts begin to pay-off since the experienced annotator becomes a potential tutor for new recruits.

To improve the annotation process, two considerations are possible. First, the annotators can be automatically evaluated on a small trial set with automatic feedback. This way, a minimum level of initial expertise is guaranteed. Afterwards, it is interesting to apply active learning strategies (Settles, 2009) to automatically select which sentences to annotate. In addition to standard active learning approaches, where a classifier-based uncertainty or informativeness measure is used, in this context the inter-annotator agreement could be directly used to re-sample sentences for which the agreement is low, so that the more complex sentences receive more annotations.

6 Conclusion

This paper presents the preliminary results of ongoing research. The majority of current research in the CORD-19 corpus uses unsupervised or semi-supervised approaches for knowledge discovery. We propose a supervised approach for extracting semantic relations and concepts in scientific articles using the CORD-19 corpus. For this purpose we annotated 500 sentences using a general purpose semantic model. We propose a baseline text-mining pipeline, trained on this data and executed on 100,959 additional sentences of the CORD-19 corpus, for automatically extracting relevant knowledge. This approach allows the discovery of relevant facts mentioned in research papers with fine-grained semantics, including causality, compositionality, and contextual dependencies. The annotated corpus and baseline implementation can be used as a starting point for developing more powerful knowledge discovery systems that can automatically analyze the growing body of scientific research related to the COVID-19 epidemic and similar future scenarios.

Acknowledgments

Funding: This research has been supported by a Carolina Foundation grant in agreement with University of Alicante and University of Havana. Moreover, the research has been partially funded by the University of Alicante and the University of Havana, the Generalitat Valenciana (Conselleria d’Educació, Investigació, Cultura i Export) and the Spanish Government through the projects LIVING-LANG (RTI2018-094653-B-C22), INTEGER (RTI2018-094649-B-I00) and SIIA (PROMETEO/2018/089, PROMETEU/2018/089). Additionally, it has been backed by the work of both COST Actions: CA19134 – Distributed Knowledge Graphs and CA19142 – Leading Platform for European Citizens, Industries, Academia and Policymakers in Media Accessibility.

This research was possible thanks to the voluntary effort of several annotators, including: Abel Molina Sánchez, Alejandro Klever Clemente, Daniel Alejandro Cárdenas Cabrera, Daniel Valdés Pérez, Gabriela Bárbara Martínez Giraldo, Gabriela Fernández Ménade, Houcemeddine Turki, Inti Antonio Blanco González, We sincerely thank all of the above and the remaining anonymous annotators. Finally, we would like to thank the anonymous reviewers. Their helpful comments have been considered in the current version of this paper.

References


Fausto Giunchiglia and Mattia Fumagalli. 2017. Tel-


Online Learning over Time in Adaptive Neural Machine Translation

Thierry Etchegoyhen∗ David Ponce∗ Harritxu Gete Ugarte Victor Ruiz Gómez
Vicomtech Foundation, Basque Research and Technology Alliance (BRTA)
{tetchegoyhen,adponce,hgete,vruiz}@vicomtech.org

Abstract

Adaptive Machine Translation purports to dynamically include user feedback to improve translation quality. In a post-editing scenario, user corrections of machine translation output are thus continuously incorporated into translation models, reducing or eliminating repetitive error editing and increasing the usefulness of automated translation. In neural machine translation, this goal may be achieved via online learning approaches, where network parameters are updated based on each new sample. This type of adaptation typically requires higher learning rates, which can affect the quality of the models over time. Alternatively, less aggressive online learning setups may preserve model stability, at the cost of reduced adaptation to user-generated corrections.

In this work, we evaluate different online learning configurations over time, measuring their impact on user-generated samples, as well as separate in-domain and out-of-domain datasets. Results in two different domains indicate that mixed approaches combining online learning with periodic batch fine-tuning might be needed to balance the benefits of online learning with model stability.

1 Introduction

Machine Translation (MT) quality has increased significantly in recent years, notably with the advent of modern Neural Machine Translation (NMT) approaches (Bahdanau et al., 2015; Vaswani et al., 2017). Despite this progress, machine translated output requires post-editing in many cases, a process which is made more taxing when the same errors are repeated by MT systems segment after segment.

To tackle this issue, adaptive approaches to machine translation aim to incorporate user feedback, oftentimes in post-editing scenarios (Turchi et al., 2017), although on-the-fly adaptation is also relevant for interactive machine translation (Peris et al., 2017). In NMT, responsive model adaptation can be achieved via online learning approaches, where network parameters are updated based on each new sample of user-edited data. To perform this type of adaption from single data points, higher optimiser learning rates (LR) are typically required, which can affect the quality of the models over time. Alternatively, less aggressive online learning setups may preserve model stability, at the cost of reduced adaptation to user-generated corrections.

In this work, we study the evolution of online learning over time in a post-editing scenario, to determine optimal configurations in terms of both adaptation to continuous user input and model stability. For this purpose, we examine the behaviour of four different gradient-descent optimisers in two different domains, with varying learning rates, and evaluate their behaviour as the number of samples increases over time.

To measure system responsiveness to user input over time, we evaluate online learning on dynamically increasing sets of samples formed by simulated user corrections. To determine model stability as online learning is performed, we also measure the quality of the MT models on static test sets pertaining to the domain at hand and on out-of-domain datasets, as additional measures of model evolution over time via online learning. Additionally, we compare the best online learning variants to models trained via batch fine-tuning on accumulated user data.

To our knowledge, this type of evaluation has not been previously explored and our results can support further work on online learning for NMT, as well as help practitioners in the field determine optimal configurations to design responsive and balanced adaptive MT systems.
2 Related Work

Most studies of online learning for machine translation have taken place in the context of Statistical Machine Translation (SMT) (Brown et al., 1990; Koehn, 2010). Several methods have thus been proposed to adapt phrase tables and language models of SMT models, in post-editing, interactive or streaming scenarios (Hardt and Elming, 2010; Ortiz-Martínez et al., 2010; Levenberg et al., 2010; Bertoldi et al., 2014). Evaluations of user productivity in adaptive SMT have notably shown a significant overall reduction of effort in post-editing scenarios (Bentivogli et al., 2015).

In Neural Machine Translation, comparatively fewer studies have been dedicated to online learning approaches. Turchi et al. (2017) explore different strategies based on a posteriori integration of human post-edits, a priori adaptation by tuning to similar sentences in the training data, and a combination of both, showing substantial improvements over static models. Peris et al. (2017) compared SMT and NMT models in interactive MT scenarios, demonstrating significant improvements in effort reduction with the latter over strong phrase-based systems. In Peris and Casacuberta (2019), online training of NMT models is evaluated under both post-editing and interactive scenarios. Similarly to the present work, they compared different optimisers with varying learning rates on datasets covering five different domains and different scenarios, assuming availability or lack of in-domain data prior to online learning. We complement their work in the present study by measuring the precise evolution of online learning over time and comparing it to batch fine-tuning at different time steps.1

Several user-centric studies have also demonstrated the usefulness of online learning for NMT, via analyses of user effort in static and adaptive environments (Karimova et al., 2018; Simianer et al., 2019; Domingo et al., 2019, 2020).

3 Experimental Setup

In this section, we describe in turn the core components of our experiments, namely the selected corpora, the training modalities of the different types of NMT models, and the selected optimisers.

3.1 Corpora

We first selected four datasets to train a generic model based on out-of-domain data.2 To mimic a typical multi-domain generic model, we selected the following corpora: Europarl (Koehn, 2005), MultiUN (Eisele and Chen, 2010), OpenSubs (Lison and Tiedemann, 2016) and CC-Align (El-Kishky et al., 2020). Each of the four corpora was downsampled to the first 1M parallel sentences and the resulting datasets merged into a unique parallel corpus (Generic), from which development and test sets were extracted via uniform sampling.

As a basis for online learning, we selected two separate domain-specific datasets. In both cases, we used publicly available datasets and followed a similar methodology: the available test sets were used as is, to measure in-domain model stability over time, and are referred to as static in-domain test sets; the first 100K of the training sets were selected to simulate user post-editing, with the reference translations taken to be the post-edited version of the translated source segments, following standard practices in experimental protocols to evaluate MT adaptation and online learning (Ortiz-Martínez, 2016; Peris and Casacuberta, 2019). We refer to these datasets as dynamic in-domain, which are used to both perform incremental training and test the models in an online learning scenario. Dynamic ID datasets are further split in gradually increasing subsets of order $10^n$, with $n \in \{0, 1, 2, 3, 4, 5\}$, starting from the first sentence.

As our first in-domain (ID) test case, we selected the TED corpus (Cettolo et al., 2012), using tst2015 as development set, tst2016 as test set and the first 100K pairs of the 2016 training set as dynamic ID set for this domain.3 As this corpus consists of first-person presentations on varied scientific or technological topics, it is markedly different from the datasets selected to train the generic domain.

We chose NewsCommentary v16 as our second in-domain dataset, in the concatenated version available on OPUS, using the first 100K pairs as online training data, the next 1522 pairs as development data, and the last 3000 as test set. This corpus consists of third-person news commentary and is thus relatively closer to the generic corpora in terms of topics and style.

1Peris and Casacuberta (2019) also include a scenario where online learning is applied over models first trained via batch fine-tuning on in-domain data, a setup which differs from our experiments.

2Unless otherwise specified, all datasets are those available on the OPUS repository (Tiedemann, 2012), as of April 2021.

3We used the version of the corpus available here: https://wit3.fbk.eu/2016-01-d

417
All corpora were tokenised and truecased with Moses scripts (Koehn, 2005) and words segmented via joint Byte Pair Encoding (Sennrich et al., 2016), using 30K merge operations. Statistics of the prepared corpora are summarised in Table 1.

### 3.2 Models

All translation models were based on the Transformer-base architecture (Vaswani et al., 2017) and built with the MarianNMT toolkit (Junczys-Dowmunt et al., 2018). The models consist of 6-layer encoders and decoders, feed-forward networks of 2048 units, embeddings vectors of dimension 512 and 8 attention heads. The dropout rate between layers is 0.1 and embeddings for the source, target and output layers were tied.

For the generic static models, we used the Adam optimiser (Kingma and Ba, 2015) with $\alpha = 0.0003$, $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. The learning rate was set to increase linearly for the first 16,000 training steps and decrease thereafter proportionally to the inverse square root of the corresponding step. We set the working memory to 6000MB and with a mini-batch set to automatically fit the specified memory. The validation data were evaluated every 3,500 steps and patience was set to 10.

For the online models, trained on the dynamic ID sets, the models were updated incrementally, with batches of one source-reference pair and a single update of the network based on the sample at hand. Each version of the model resulting from an update as described was taken as the basis for the next online update. We selected four representative optimisers, described in the next section.

To measure the impact of domain variation, all models were trained for translation from English to Spanish. Evaluation was performed on the BLEU metric (Papineni et al., 2002), computed with the sacreBLEU toolkit (Post, 2018).6

### 3.3 Optimisers

Stochastic gradient descent (SGD) (Robbins and Monro, 1951) is one of the most common methods to estimate the parameters of a network, given the gradient of an error function, as it computes estimates on a per-sample basis. In NMT, SGD usually takes the form of mini-batch SGD, where the gradient is computed as the average gradient of the samples in a mini-batch; we use SGD as a shortcut for mini-batch SGD in what follows.

Several optimisations have been proposed to address some limitations of SGD, in particular methods that include a parameter-level update of the learning rate. Among these approaches, Adagrad (Duchi et al., 2011) uses past gradients for each parameter to compute parameter-level updates. Adadelta (Zeiler, 2012) extends it by mainly restricting the accumulation of past gradients to a fixed window size, an approach independently proposed as the basis of the RMSProp optimiser. Another popular alternative is Adam (op. cit), which includes bias-corrected estimates of the 1st and 2nd moment, and is the default optimiser to train Transformer models in toolkits such as MarianNMT.

As indicated in Section 2, previous studies in online learning for NMT have compared the aforementioned parameter update methods, reaching different conclusions. Thus, Turchi et al. (2017) concluded that vanilla SGD was the optimal optimiser overall in their experiments, where the learning rate was fixed to 0.001 for all optimisers, whereas Peris and Casacuberta (2019) reached the conclusion that Adadelta, and to a lesser degree, SGD, were optimal after selecting the learning rate for each optimiser separately via grid-search on development sets. To gain further insights on optimal configurations for online learning, we selected four of the main optimisers, namely SGD, Adam, Adagrad and RMSProp, and measured the impact of different learning rates at different points in time, as described in the next section.7

---

6Adadelta can also be computed with a second type of estimate, using past updates instead of Lasso regularisation. To limit our experiments to the main optimiser variants, we only considered the first update rule in Adadelta and refer to it as RMSProp, to avoid confusion over which version of the Adadelta updates is used.

7The implementation of RMSProp in MarianNMT is our own; all others are based on the toolkit default implementation.
To evaluate the impact of online learning over time, we first evaluate the models against the references of the dynamic in-domain training set after $10^n$ online learning updates with each of the selected optimisers, with $n \in \{0, 1, 2, 3, 4, 5\}$.

### 4.1 Impact of Learning Rate

An important component of parameter update is the learning rate, which determines the amplitude of the updates. For online learning, a critical choice needs to be made between aggressive and conservative updates, based on high or low learning rates, respectively. The former may provide rapid adaptation, at the risk of deteriorating the network from overfitting to the samples, whereas the latter may delay or dilute the expected model adaptation, thus reducing the positive impact of online learning.

To measure both extremes, we selected the best learning rates for each optimiser according to the BLEU scores obtained on the dynamic datasets on the first (best@1) and last samples (best@100K), for each of the two selected domains. To determine an optimal learning rate for the first update, we randomly sampled 100 sentences and computed the BLEU score with all learning rates variants on the dynamic ID set, selecting the learning rate with the best average BLEU. This was meant to limit the impact of the characteristics of the first sentence in these datasets, which might not be representative of the data distribution. For the last update, all data points were considered to determine the best BLEU scores and associated learning rate.

Figure 1 and Figure 2 show the evolution of the optimisers for TED and NewsCommentary, respectively. We also include the evolution of the baseline generic models and models trained via batch fine-tuning over the available data (1 and 100K samples
Figure 3: BLEU scores as a function of number of samples and learning rates on the TED corpus

Figure 4: BLEU scores as a function of number of samples and learning rates on the NewsCommentary corpus
for figures (a) and (b), respectively); in the latter case, the optimiser is the one used to train the baselines, namely Adam, with the learning rate and moment inherited from the last baseline update.

Adam displayed a more erratic behaviour than the other optimisers, with sharp degradation after 10K updates when selecting a conservative learning rate, and after 100 with the more aggressive LR. Except for TED with the best \(0.1\) LR, where all optimisers started with maximal adaptation on the first update, Adagrad performed worse initially but eventually converged with SGD and RMSProp, while also obtaining the highest scores between 10 and 10K samples on TED with its most conservative LR, and between 10 and 1000 on NewsCommentary. SGD and RMSProp behave similarly with a more stable behaviour, except on NewsCommentary where RMSProp performed markedly better with a conservative LR. It is also worth noting that the best LRs, whether aggressive or conservative, differ in most cases depending on the domain. Although this might be expected considering that the selected domains differ in terms of proximity to the generic data, as previously noted, these differences illustrate the delicate task of determining optimal online learning setups across the board.

The baseline evolved as expected, with lower scores than models benefitting from in-domain data. For batch fine-tuning, the evolution featured increasing scores as more data are available, eventually converging with the best optimiser variants. Overall, all variants of online learning tended towards degraded performance as the number of samples and subsequent updates increased, particularly with high learning rates. This is not unexpected given the overfitting associated with network adaptation over minimal samples, but both SGD and RMSProp appeared beneficial at least up to the 100K mark on the dynamic in-domain sets. We will examine the behaviour of all models on the static and out-of-domain test sets in Section 5.

4.2 Optimal Optimiser Setup

So far we have examined the behaviour of the different optimisers over time with the best LR at the two extremes, i.e. for 1 and 100K samples. To determine whether other learning rates might be optimal at other time steps, we computed BLEU scores on the dynamic in-domain set as a function of both learning rates and number of samples. Figure 3 and Figure 4 show the results on TED and NewsCommentary, respectively.

With Adam, other learning rates are more stable over time than the best performing one selected on the basis of a single (averaged) sample, with higher scores and less erratic behaviour, in particular on NewsCommentary. Nonetheless, even these more balanced learning rates achieve poorer scores than the other three optimisers overall, for both initial and final updates.

For Adagrad, other values than the ones based on the extremes performed better for some intermediate sample subsets on the TED dataset, but the more aggressive LR was optimal overall on NewsCommentary, achieving better scores than all other optimisers as the number of samples increased.

RMSProp also achieved an overall better distribution of scores when selecting the most aggressive LR on TED. On NewsCommentary, the selected best LR for the initial sample \( (0.1) \) was not the optimal choice, although it performed closely to the optimal \(0.05\). Note that both LRs achieve an identical score on the first sample, but on the average score obtained on the 100 randomly selected sentences used to to select the most aggressive LR, the previously selected LR of \(0.1\) was markedly better.

Similarly, for SGD on NewsCommentary, the best LR option over the averaged 100 unique samples \( (0.1) \) performed slightly worse overall than an LR of \(0.05\) as the number of samples increased; on TED, the most aggressive SGD LR performed better than the alternatives, except when the number of samples reached the 100K mark.

Although these results show that selecting an optimal learning rate for either optimiser is bound to be less than optimal at a given point in time, both SGD and RMSProp with an LR of \(0.05\) appear to be reasonable choices that provide overall benefits on the two selected datasets. Interestingly, this value differs from the optimal ones established for SGD in separate experiments by Turchi et al. (2017) and Peris and Casacuberta (2019) (see Section 3.3), showing that LR selection for online learning might be dependent on domains and datasets.

5 Model Stability Over Time

As described in the previous sections, online learning can support post-editing by adapting to user corrections incrementally. This is obtained via relatively aggressive learning rates that enable updates to be significant on the basis of unique training
Figure 5 shows the results for TED and NewsCommentary at the 100K mark, when selecting the best learning rate for each optimiser based on the best BLEU scores for the initial updates, which, as a reminder, were computed over random samples of 100 sentences. On TED, batch fine-tuning outperformed all variants of online learning on the in-domain datasets, both static and dynamic, although only slightly over Adagrad on the static ID test set. On NewsCommentary, Adagrad outperformed all variants on the dynamic ID dataset, while also being only slightly under batch fine-tuning on the static ID test set. However, as described in previous sections, this optimiser also performed significantly worse than SGD and RMSProp for initial updates, thus being less beneficial for initial stages of online learning. When compared to the most efficient optimisers for earlier online learning, namely SGD and RMSProp, batch fine-tuning would be the favoured option when reaching at least 100K data points.

Figure 6 presents the results after the final update stage when taking the best learning rates at the 100K mark for all optimisers. In this scenario, on the TED datasets all optimisers except Adam feature results that are closer to those achieved via batch fine-tuning, although the latter obtained better results overall on both the dynamic and the static in-domain datasets. On NewsCommentary, SGD, RMSProp and Adagrad significantly outperformed batch fine-tuning on the dynamic data, while the latter performed slightly better on the static ID test set but with minor differences. Among optimisers in online learning scenarios, SGD would be favoured when selecting more conservative learning rates, although both RMSProp and SGD would
also be favoured over batch fine-tuning in such a scenario for NewsCommentary. However, a more conservative learning rate also reduces its benefits at earlier stages, and the single case where online learning optimisers outperform batch fine-tuning at later stages might not be relevant in actual usage.

In terms of out-of-domain data, batch fine-tuning performed better in all but one case, namely TED with the best learning rate for initial updates, where Adagrad performed better. Batch fine-tuning performed similarly to the baseline on NewsCommentary, which may be attributed to the relative proximity of NewsCommentary data to the generic training data, and conversely to the higher data difference between TED and the datasets that compose the generic training sets.

6 Conclusions

In this paper, we explored the behaviour of online learning for Neural Machine Translation over time, examining the results obtained with four different optimisers as the number of samples increases and evaluating translation model evolution after repeated network updates with different types of learning rates, from most aggressive to most conservative. We also compared online learning with batch fine-tuning on dynamic and static datasets, as well as out-of-domain test sets, to measure overall model stability.

On the two domains we explored, based on TED and NewsCommentary data, there does not appear to be an optimal configuration, where online learning would be optimal in both the short and long term. SGD and RMSProp both feature a learning rate value which provides early benefits of online learning with relatively minor degradation over time, and might be viewed as the most balanced configuration in our experiments.

However, at least in the domains we explored, batch fine-tuning was shown to be preferable at later stages in terms of model stability across dynamic, static and out-of-domain datasets. For practical adaptive NMT, it might thus be preferable to combine online learning, over limited time steps, with periodic batch fine-tuning over previous model checkpoints on the data accumulated over time.
Acknowledgments

This work was partially supported by the Department of Economic Development of the Basque Government (Spri), as part of project ITAI (ZL-2021/00038). We would like to thank the anonymous RANLP reviewers for their insightful comments and useful feedback.

References


Improving Character-Aware Neural Language Model by Warming Up Character Encoder under Skip-gram Architecture

Yukun Feng¹, Chenlong Hu¹, Hidetaka Kamigaito¹, Hiroya Takamura² and Manabu Okumura¹

¹Tokyo Institute of Technology
²National Institute of Advanced Industrial Science and Technology (AIST)
{yukun, huchenlong, kamigaito}@lr.pi.titech.ac.jp
oku@pi.titech.ac.jp takamura.hiroya@aist.go.jp

Abstract

Character-aware neural language models can capture the relationship between words by exploiting character-level information and are particularly effective for languages with rich morphology. However, these models are usually biased towards information from surface forms. To alleviate this problem, we propose a simple and effective method to improve a character-aware neural language model by forcing a character encoder to produce word-based embeddings under Skip-gram architecture in a warm-up step without extra training data. We empirically show that the resulting character-aware neural language model achieves obvious improvements of perplexity scores on typologically diverse languages, that contain many low-frequency or unseen words.

1 Introduction

Neural language models (NLM) usually maintain a fixed vocabulary and map each word to a continuous representation. These models cannot handle new words and are not effective for languages with rich morphology. One solution is to use smaller units, such as bytes, characters, or word pieces learned from word tokens (Wu et al., 2016; Sennrich et al., 2016). However, this approach has to process longer sequences than word-level alternatives and may increase modeling and computational challenges (Cherry et al., 2018). This paper focuses on another way based on word-level models in which a character encoder is used on top of characters of each word to calculate the word representation. They are often referred to as character-aware NLMs (CNLMs) (Ling et al., 2015; Kim et al., 2016; Vania and Lopez, 2017; Gerz et al., 2018; Assylbekov and Takhanov, 2018; Feng et al., 2019). However, the character encoders in CNLMs often show over-representation of orthography rather than semantic meaning in the resulting word embedding despite the fact that training word-based NLMs usually helps learn such semantic meaning (Kim et al., 2016; Vania and Lopez, 2017; Assylbekov and Takhanov, 2018). For example, in CNLMs, the nearest neighbors of the word ‘his’ with cosine similarity are ‘hhs’ and ‘this’ while ‘my’ is far from the nearest neighbors.

To alleviate the over-representation issue in CNLMs, we propose to directly force the character encoder to produce the word-based embedding in a warm-up step before the training of CNLMs starts. Specifically, the character encoder encodes an input word, and the encoded embedding will be forced to be close to the embeddings of its surrounding words and far from the word embeddings of negative samples. Unlike the dynamically constructed embedding of the input word, the embeddings of surrounding words and negative samples are word-based, and thus these embeddings will not be biased to surface forms. The above method is similar to the architecture of the Skip-gram model (Mikolov et al., 2013) with the difference that we use a complex character encoder which is shown to be powerful for languages with rich morphology.

In our experiments¹, we choose the widely used long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and the recent state-of-the-art AWD-LSTM-LM (Merity et al., 2018) for language modeling. For the character encoder, we experiment with bidirectional LSTM (BiLSTM) over character trigrams as this variant has shown better performance than other character encoders on 10 languages (Vania and Lopez, 2017). We evaluate our method in two types of datasets. One contains 14 typologically diverse languages with a large number of low-frequency words and unseen words in the test set. Thus, we can test our method in a real LM setup. Another one contains 5 lan-

¹Our code can be obtained from https://github.com/yukunfeng/warmup_char_lm
guages, where the new words in the test set have been replaced to <UNK>. It is commonly used for evaluating CNLMs in this field.

Our experiments empirically show our method can achieve obviously improved perplexity scores on a wide range of languages. Finally, we analyze the learned word embedding with our character encoder on English word similarity tasks.

2 Related Work

A lot of work has tried to improve CNLMs in recent years, such as analyzing performance of CNLMs with different character encoders and character units (Vania and Lopez, 2017), reusing subword embeddings in CNLMs (Assylbekov and Takanov, 2018), injecting subword-level information into softmax (Gerz et al., 2018), and combining word- and character-level information in CNLMs (Miyamoto and Cho, 2016; Verwimp et al., 2017; Kang et al., 2011; Feng et al., 2019).

As for alleviating the over-representation issue mentioned above, Kim et al. (2016) used a highway network on top of their character encoder and found their highway network can encode semantic features that are not discernible from orthography alone. Assylbekov and Takanov (2018) used syllables and morphemes in a word to construct word embeddings and showed syllable- or morpheme-based CNLMs are less biased towards surface forms than a standard CNLM. However, this approach relies on extra toolkits to extract syllables or morphemes. To our knowledge, there is not much work particularly paying attention to this issue in CNLMs. It is discussed only in a section in the above mentioned work, and the experiment is limited to several languages. Furthermore, the analysis of character encoders is done by manually selecting several words with their nearest neighbors based on cosine similarity, while we formally verify that the character encoder captures more semantic features on 5 English word similarity tasks. Our method is simple and different from the highway network used by Kim et al. (2016). We do not need to change the existing architecture of CNLMs.

Another related work to ours is word representation learning as we utilize the Skip-gram architecture. One goal of this field is to learn word embeddings on large-scale corpus and use them on downstream tasks, which is different from ours. Our method works without extra training data, and we do not aim at transfer learning with other training data like a standard Skip-gram model.

3 Model Description

The whole architecture is shown in Figure 1. We use BiLSTM over character trigrams as our character encoder since this variant performed best on most datasets (Vania and Lopez, 2017). Given a word \( w_t \), we denote its embedding as \( x_t \in \mathbb{R}^d \), where \( d \) is the embedding size. We compute the representation of \( w_t \) in BiLSTM as follows:

\[
x_t = W_f h_{fw}^t + W_b h_{bw}^t + b,
\]

where \( h_{fw}^t, h_{bw}^t \in \mathbb{R}^d \) are the last states of the forward and backward LSTMs, respectively. \( W_f, W_b \in \mathbb{R}^{d\times d} \) and \( b \in \mathbb{R}^d \) are trainable parameters.

We adopt the basic architecture of Skip-gram model for warming up our character encoder. Given an input word \( w_t \) which will be encoded by our character encoder, we then use the encoded embedding to predict a set of output words that surround the input word in a given window. For example, when the window size is 2, the output words are \( w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2} \). We use \( o_t \) to denote the embedding of an output word for \( w_t \).

The input word embedding \( x_t \) of \( w_t \) is computed with Eq. 1. Note that \( o_t \) is word-based and thus will be not biased to its surface form, which is different from \( x_t \). Given a single training example \((w_t, w_{t+j})\), we maximize the objective function:

\[
\log \sigma(x_t^T o_{t+j}) + \sum_{i=1}^{k} \log \sigma(-x_t^T o_{t+i}),
\]

where \( k \) is the size of the negative samples, and \( \sigma \) is the sigmoid function.

After warming up, we use the trained character encoder to initialize the one in our CNLM and then train our CNLM with a standard LM loss.

4 Experiments

4.1 Datasets

We can find common language modeling datasets for evaluating CNLMs in the work of Botha and Blunsom (2014). While these datasets contain languages with rich morphology, they have only 5 different languages. The most large-scale language modeling datasets are from the work of Gerz et al. (2018), who released 50 language modeling datasets covering typologically diverse languages. The difference of the newly released datasets from
the previously common datasets is that many unseen words are kept in the test set. Thus, on the datasets, we can test our methods in a real LM setup. These languages were selected to represent a wide spectrum of different morphological systems and have a large number of low-frequency or unseen words. Thus, these datasets are desirable for checking the performance of CNLMs. Due to the large number of experiments, we chose datasets of only 14 languages from these datasets and tried to cover different language typologies as well as different type/token ratios (TTRs). The statistics of our chosen datasets are shown in Table 1.

To compare with other models, we also set up above mentioned 5 common non-English LM datasets with rich morphology from the 2013 ACL Workshop on Machine Translation, which have been commonly used for evaluating CNLMs (Botha and Blunsom, 2014; Kim et al., 2016; Bojanowski et al., 2017; Assylbekov and Takhanov, 2018; Feng et al., 2019). Note that the new words in the test set have been replaced with special <UNK>, which is not a practical setting. The data statistics is in Table 2.

4.2 Models

The hyperparameters of our LSTM language model are shown in Table 3. The learning rate was decreased if no improvement is observed in the validation set. We trained the Skip-gram architecture in warm-up step for 7 epochs with 5 negative samples for all datasets. We define Char-BiLSTM-LSTM as our CNLM, and Warmed-Char-BiLSTM-LSTM as our CNLM with a warmed character encoder.

To check our idea with a stronger baseline, we used the recent state-of-the-art AWD-LSTM-LM codebase2(Merity et al., 2018). We replaced the word embedding layer of this model with our BiLSTM character encoder. We refer to it as Char-BiLSTM-AWD-LSTM and the warmed one as Warmed-Char-BiLSTM-AWD-LSTM. Due to time constraints, we set the training epoch on all datasets to 200. We refer to the original AWD-LSTM which is word-level as Word-AWD-LSTM. For the other parameters, we followed the setting in the source code. We make sure that all models under our chosen epochs are trained to convergence so that the gain from our method is not due to longer training in warm-up.

4.3 Results on 14 Languages

The results on 14 languages are shown in Table 4. Our Char-BiLSTM-LSTM baseline outperforms Char-CNN-LSTM from (Gerz et al., 2018) on all datasets. It is also shown that as the TTR increases, Char-BiLSTM-AWD-LSTM achieves a better result than Word-AWD-LSTM. One reason may be that higher TTR languages have more low-frequency words and unseen tokens, as shown in Table 1. Thus, utilizing character information is important in these languages. Our proposed Warmed-Char-BiLSTM-LSTM and Warmed-Char-BiLSTM-AWD-LSTM achieves further obvious improvements compared with Char-BiLSTM-LSTM and Char-BiLSTM-AWD-LSTM respectively on most datasets without extra training.

---

2https://github.com/salesforce/awd-lstm-lm
Table 1: The statistics of our language modeling datasets. TTR represents the type/token ratio.

<table>
<thead>
<tr>
<th>Dataset (Language)</th>
<th>Typology</th>
<th>TTR</th>
<th>Vocab</th>
<th>#Train tokens</th>
<th>#Test tokens</th>
<th>Freq≤15 (Train)</th>
<th>#Unseen tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>zh (Chinese)</td>
<td>Isolating</td>
<td>0.06</td>
<td>43672</td>
<td>746K</td>
<td>56.8K</td>
<td>16%</td>
<td>2132</td>
</tr>
<tr>
<td>ja (Japanese)</td>
<td>Agglutinative</td>
<td>0.06</td>
<td>44863</td>
<td>729K</td>
<td>54.6K</td>
<td>15%</td>
<td>2558</td>
</tr>
<tr>
<td>pt (Portuguese)</td>
<td>Fusional</td>
<td>0.07</td>
<td>56167</td>
<td>780K</td>
<td>59.3K</td>
<td>17%</td>
<td>2947</td>
</tr>
<tr>
<td>en (English)</td>
<td>Fusional</td>
<td>0.07</td>
<td>55521</td>
<td>783K</td>
<td>59.5K</td>
<td>17%</td>
<td>3618</td>
</tr>
<tr>
<td>es (Spanish)</td>
<td>Fusional</td>
<td>0.08</td>
<td>60196</td>
<td>781K</td>
<td>57.2K</td>
<td>18%</td>
<td>3486</td>
</tr>
<tr>
<td>he (Hebrew)</td>
<td>Introflexive</td>
<td>0.12</td>
<td>83217</td>
<td>717K</td>
<td>54.6K</td>
<td>27%</td>
<td>4855</td>
</tr>
<tr>
<td>de (German)</td>
<td>Fusional</td>
<td>0.12</td>
<td>80741</td>
<td>682K</td>
<td>51.3K</td>
<td>24%</td>
<td>5451</td>
</tr>
<tr>
<td>ar (Arabic)</td>
<td>Introflexive</td>
<td>0.12</td>
<td>89089</td>
<td>722K</td>
<td>54.7K</td>
<td>26%</td>
<td>6076</td>
</tr>
<tr>
<td>cs (Czech)</td>
<td>Fusional</td>
<td>0.14</td>
<td>86783</td>
<td>641K</td>
<td>49.6K</td>
<td>30%</td>
<td>5436</td>
</tr>
<tr>
<td>ru (Russian)</td>
<td>Fusional</td>
<td>0.15</td>
<td>98097</td>
<td>666K</td>
<td>48.4K</td>
<td>32%</td>
<td>4881</td>
</tr>
<tr>
<td>et (Estonian)</td>
<td>Agglutinative</td>
<td>0.17</td>
<td>94184</td>
<td>556K</td>
<td>38.6K</td>
<td>34%</td>
<td>4960</td>
</tr>
<tr>
<td>fi (Finnish)</td>
<td>Agglutinative</td>
<td>0.2</td>
<td>115579</td>
<td>585K</td>
<td>44.8K</td>
<td>38%</td>
<td>7899</td>
</tr>
<tr>
<td>ko (Korean)</td>
<td>Agglutinative</td>
<td>0.22</td>
<td>143794</td>
<td>648K</td>
<td>50.6K</td>
<td>42%</td>
<td>9745</td>
</tr>
<tr>
<td>kn (Kannada)</td>
<td>Agglutinative</td>
<td>0.22</td>
<td>94660</td>
<td>434K</td>
<td>29.4K</td>
<td>41%</td>
<td>5214</td>
</tr>
</tbody>
</table>

Table 2: The statistics of our 5 language modeling datasets.

<table>
<thead>
<tr>
<th>Vocab size</th>
<th>#Train token</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech (CS)</td>
<td>46K</td>
</tr>
<tr>
<td>German (DE)</td>
<td>37K</td>
</tr>
<tr>
<td>Spanish (ES)</td>
<td>27K</td>
</tr>
<tr>
<td>French (FR)</td>
<td>25K</td>
</tr>
<tr>
<td>Russian (RU)</td>
<td>86K</td>
</tr>
</tbody>
</table>

Table 3: Hyperparameters of our model. We use \(d\) for the size of the character/word embeddings and for the number of hidden units of LSTM and Bi-LSTM.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding size (d)</td>
<td>650</td>
</tr>
<tr>
<td>LSTM layers</td>
<td>2</td>
</tr>
<tr>
<td>LSTM sequence length</td>
<td>35</td>
</tr>
<tr>
<td>Param. init: rand uniform</td>
<td></td>
</tr>
<tr>
<td>Dropout</td>
<td>0.5</td>
</tr>
<tr>
<td>Epochs</td>
<td>40</td>
</tr>
<tr>
<td>Optimizer</td>
<td>SGD</td>
</tr>
<tr>
<td>Learning rate</td>
<td>20</td>
</tr>
<tr>
<td>Learning rate decay</td>
<td>4</td>
</tr>
<tr>
<td>Gradient clipping</td>
<td>0.25</td>
</tr>
<tr>
<td>Batch size</td>
<td>20</td>
</tr>
</tbody>
</table>

4.4 Results on 5 Common Datasets

The results on common datasets are shown in Table 5. Most work aims at improving CNLMs at different aspects and the gain comes from different new information. For example, the gain of CNLM from Feng et al. (2019) comes from injecting word-level information into CNLM, and Assylbekov and Takhanov (2018) improved CNLMs by using morphemes and reusing weights. Bojanowski et al. (2017) used conventional word-level LSTM-LM instead of CNLM, and their goal is not to improve CNLMs. The gain from their model comes from transferring word embeddings learned through Skip-gram that considers character-level information to word-level LSTM-LM without character-level information. That is, their method used new information for their LSTM-LM while in our method there is no extra new information. As shown in Table 5, our baseline model is strong compared with most models, and our method can further improve it without extra new information.

5 Analysis

5.1 Analysis of Character Encoder

Unlike prior work which analyzes their character encoder by manually selecting several words and their nearest neighbors based on cosine similarity, we formally verify whether our method helps the character encoder capture more semantic features on English word similarity tasks. We chose the English dataset ‘en’ shown in Table 1 as our training set. Specifically, after finishing the training of
Table 4: Perplexity results for our models and several baselines.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>zh</th>
<th>ja</th>
<th>pt</th>
<th>en</th>
<th>es</th>
<th>he</th>
<th>de</th>
<th>ar</th>
<th>cs</th>
<th>ru</th>
<th>et</th>
<th>fi</th>
<th>ko</th>
<th>kn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Char-CNN-LSTM</td>
<td>797</td>
<td>136</td>
<td>214</td>
<td>371</td>
<td>275</td>
<td>1519</td>
<td>602</td>
<td>1659</td>
<td>1252</td>
<td>812</td>
<td>1478</td>
<td>2236</td>
<td>4778</td>
<td>2558</td>
</tr>
<tr>
<td>Char-BiLSTM-LSTM</td>
<td>578</td>
<td>107</td>
<td>178</td>
<td>302</td>
<td>230</td>
<td>1170</td>
<td>483</td>
<td>1337</td>
<td>973</td>
<td>620</td>
<td>967</td>
<td>1648</td>
<td>3247</td>
<td>1543</td>
</tr>
<tr>
<td>Warmed-Char-BiLSTM-LSTM</td>
<td>480</td>
<td>99</td>
<td>162</td>
<td>278</td>
<td>208</td>
<td>1005</td>
<td>439</td>
<td>1158</td>
<td>843</td>
<td>503</td>
<td>877</td>
<td>1435</td>
<td>2472</td>
<td>1724</td>
</tr>
<tr>
<td>Word-AWD-LSTM</td>
<td>481</td>
<td>98</td>
<td>165</td>
<td>289</td>
<td>208</td>
<td>1005</td>
<td>439</td>
<td>1158</td>
<td>843</td>
<td>503</td>
<td>877</td>
<td>1435</td>
<td>2472</td>
<td>1724</td>
</tr>
<tr>
<td>Char-BiLSTM-AWD-LSTM</td>
<td>497</td>
<td>99</td>
<td>156</td>
<td>278</td>
<td>205</td>
<td>1005</td>
<td>439</td>
<td>1158</td>
<td>843</td>
<td>503</td>
<td>877</td>
<td>1435</td>
<td>2472</td>
<td>1724</td>
</tr>
<tr>
<td>Warmed-Char-BiLSTM-AWD-LSTM</td>
<td>414</td>
<td>87</td>
<td>136</td>
<td>236</td>
<td>183</td>
<td>236</td>
<td>878</td>
<td>971</td>
<td>718</td>
<td>485</td>
<td>768</td>
<td>1278</td>
<td>2082</td>
<td>1271</td>
</tr>
</tbody>
</table>

Table 5: Perplexity of our models and previous work.

<table>
<thead>
<tr>
<th>Model</th>
<th>CS</th>
<th>DE</th>
<th>ES</th>
<th>FR</th>
<th>RU</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLBL (Botha and Blunsom, 2014)</td>
<td>465</td>
<td>296</td>
<td>200</td>
<td>225</td>
<td>304</td>
</tr>
<tr>
<td>MorphSum (Kim et al., 2016)</td>
<td>398</td>
<td>263</td>
<td>177</td>
<td>196</td>
<td>271</td>
</tr>
<tr>
<td>CharCNN (Kim et al., 2016)</td>
<td>371</td>
<td>239</td>
<td>165</td>
<td>184</td>
<td>261</td>
</tr>
<tr>
<td>SkipGram initialization (Bojanowski et al., 2017)</td>
<td>312</td>
<td>206</td>
<td>145</td>
<td>159</td>
<td>206</td>
</tr>
<tr>
<td>MorphSum+RE+RW (Assylbekov and Takhanov, 2018)</td>
<td>338</td>
<td>222</td>
<td>157</td>
<td>172</td>
<td>210</td>
</tr>
<tr>
<td>Word-Char-LSTM (Feng et al., 2019)</td>
<td>287</td>
<td>192</td>
<td>135</td>
<td>152</td>
<td>201</td>
</tr>
<tr>
<td>Char-BiLSTM-LSTM</td>
<td>311</td>
<td>198</td>
<td>144</td>
<td>164</td>
<td>223</td>
</tr>
<tr>
<td>Warmed-Char-BiLSTM-LSTM</td>
<td>290</td>
<td>190</td>
<td>134</td>
<td>150</td>
<td>203</td>
</tr>
</tbody>
</table>

Table 6: Results on word similarity datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Word pairs</th>
<th>Char-BiLSTM-LSTM</th>
<th>Warmed-Char-BiLSTM-LSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEN</td>
<td>3000</td>
<td>10.55</td>
<td>12.52</td>
</tr>
<tr>
<td>MTurk287</td>
<td>287</td>
<td>24.47</td>
<td>26.84</td>
</tr>
<tr>
<td>MTurk771</td>
<td>771</td>
<td>3.06</td>
<td>8.59</td>
</tr>
<tr>
<td>RW</td>
<td>2034</td>
<td>17.30</td>
<td>18.85</td>
</tr>
<tr>
<td>WS353</td>
<td>353</td>
<td>15.17</td>
<td>18.54</td>
</tr>
</tbody>
</table>

For example, we calculated the perplexity of the next word, when a rare word, whose frequency is less than 15, is given as the current word. A similar analysis on language models can be found in Vania and Lopez (2017). For simplicity, we only choose the English dataset ‘en’ and the German dataset ‘de’. To fairly compare with Word-LSTM, our analysis does not contain new words in the test data. As we see in Table 7, Char-BiLSTM-LSTM mainly obtained improvements on rare word group compared with Word-LSTM. When warmed up, Char-BiLSTM-LSTM obtained further improvements both on frequent and rare word groups. Note that the reason of the gain is not that the warm
We would like to thank anonymous reviewers for their constructive comments and Hu also thanks the support from China Scholarship Council.

Acknowledgments

References


Abstract

With the increasing adoption of technology, more and more systems become target to information security breaches. In terms of readily identifying zero-day vulnerabilities, a substantial number of news outlets and social media accounts reveal emerging vulnerabilities and threats. However, analysts often spend a lot of time looking through these decentralized sources of information in order to ensure up-to-date countermeasures and patches applicable to their organisation’s information systems. Various automated processing pipelines grounded in Natural Language Processing techniques for text classification were introduced for the early identification of vulnerabilities starting from Open-Source Intelligence (OSINT) data, including news websites, blogs, and social media. In this study, we consider a corpus of more than 1600 labeled news articles, and introduce an interpretable approach to the subject of cyberthreat early detection. In particular, an interpretable classification is performed using the Longformer architecture alongside prototypes from the ProSENet structure, after performing a preliminary analysis on the Transformer’s encoding capabilities. The best interpretable architecture achieves an 88% F2-Score, arguing for the system’s applicability in real-life monitoring conditions of OSINT data.

1 Introduction

With the increasing number of cybersecurity attacks, institutions need to join forces for the prevention and detection of cyberthreats. Malicious entities target companies, but also individuals and public institutions (including health organisations), with an overall expected cost of 6000 bn $ per year (Morgan, 2020). For example, the number of requests on the French malware assistance plat-
form has tripled between 2018 and 2019\(^1\), and this trend has worsened during the pandemic (Pinhasi and Huseman, 2021) due to the increased use of technology, the advent of remote working, coupled with an exponential growth of IoT. In their endless race against hackers, security experts need to detect zero-day vulnerabilities and understand new methods employed by hackers to exploit them (Lewis, 2018). In addition, both institutions and individuals need localized expertise applicable for their environment, given its specificity in terms of employed technologies.

Large companies often have their dedicated entity, the Security Operations Center (SOC), whose experts survey Open-Source Intelligence (OSINT) data to identify new emerging vulnerabilities (Dionísio et al., 2019); however, lower-sized entities and individuals should have this information also readily available. In addition, each data source (and there are a lot of possible venues – e.g., more than 40 major blogs and newspapers are reported by Feedspot (2021)) has its own targeted fields and technologies of main interest, as well as its subjectivity in expressing the breath and impact of the vulnerability. Manual searches in multiple sources is an overall tedious process, and the wide scattering of news feeds makes the day-by-day surveillance a daunting task; as such, an automated detection of zero-day vulnerabilities from OSINT data becomes a necessity (Le Sceller et al., 2017).

Our goal is to provide an automated filtering of daily news feeds to identifying emergent cybersecurity threats as an initial screening for security experts. We emphasize from the beginning the importance of recall, namely it is critical not to disregard potential threats. Moreover, model interpretability is an important dimension of the analysis in order to provide a preliminary grounding for the model’s decisions. Hence, our research question is: To what extent can automated systems detect cybersecurity threats in news articles while ensuring interpretable results?

The paper is structured as follows. The second section introduces related work on real-time identification of threats and interpretable Natural Language Processing (NLP) approaches, while the third section introduces our approach, composed of a deep analysis of the context and the design of an interpretable model. The fourth section presents results alongside performance metrics, followed by discussions and conclusions.

## 2 Related Work

In this section, we review relevant related research on real-time threat identification and interpretable models in NLP. The abundance of OSINT data brought by Twitter has enabled SOCs to develop cyber-threat intelligence with increasing performance, while exploring tweets in different manners (e.g., by CVE(Sabotke et al., 2015) or by account(Dionísio et al., 2019)). Nonetheless, to our knowledge, existing studies focus on performance and do not consider interpretability, which seems crucial for such a critical task.

### 2.1 Real-time Threat Identification

Several papers (Attarwala et al., 2017; Dionísio et al., 2019; Le Sceller et al., 2017) introduced Twitter-based approaches to design a pipeline for threat detection and, more generally, semantic analysis. Dionísio et al. (2019) start from a set of customers, whose experts chose the Twitter cybersecurity accounts to monitor. From these accounts, tweet texts are collected using a keyword-based selection. They rely on a Convolutional Neural Network (CNN) to select only interesting tweets, while also performing named entity recognition, and check performance using True Positive and Negative Rate. A comparison between the dates from first tweet disclosing the threat to the release on the national vulnerability database of a CVSS provides insights on the ability of Twitter to become an efficient cyberthreat detection platform.

Le Sceller et al. (2017) query tweets based on keywords. Their aim is to detect and characterize cybersecurity events using only texts from tweets. A preliminary taxonomy is built to understand how main cybersecurity keywords interact, when taking decisions. The collected texts are embedded using TF-IDF and a clustering algorithm allows for up-to-date unsupervised grouping of tweets. The clustering is applied in a dynamic manner to avoid obscelence, as the keyword search is adapted according to relations and co-occurrences of words from tweets considered interesting upstream. These new keywords for the search are proposed to an external human actor, who takes the decision of adding them or not. Thus, the up-to-dateness of the architecture is guaranteed manually by experts, who

---

\(^1\)https://www.cybermalveillance.gouv.fr/tous-nos-contenus/actualites/rapport-activite-2020
leave a trace of their analysis of new trends within the model.

Abdullah et al. (2018) directly use data from new articles to detect cyberthreats. After crawling various news websites, the authors define manually, together with experts, a certain number of features defining cyberthreats, such as the threat actors, the name of the cyberattack, or the jeopardized domain. The authors create a dictionary for each feature, in which occurrences of each word corresponding to this feature are stored with their context. Using this dictionary, a Conditional Random Field enables the detection of cyberattacks from sentences. Finally, Latent Semantic Analysis supports the categorization of articles, following the type of cyberattack they shed light upon.

2.2 Interpretability in NLP

Two classes of methods tackling interpretability coexist in the literature, namely: a) a posteriori methods, which take as input a model as such, and try to explain its decisions, and b) interpretable architectures whose interpretability is taken into account in their design.

With regards to a posteriori methods, several papers (Ribeiro et al., 2016; Sundararajan et al., 2017) have studied ways to determine and visualize the isolated influence of each variable from the input of a model. Ribeiro et al. (2016) designed LIME (Local Interpretable Model-agnostic Explanations), a tool which enables the local visualization (i.e., in the input space) of the influence of each interpretable component on the decision. The model whose decision we want to explain is locally approximated by a simple-and thus interpretable-classifier. In order to make these local explanations global, a fixed number of explanations are chosen a using Submodular Pick to render, as well as possible, the use of the features by the algorithm. These explanations are then aggregated into a global result. They also introduce metrics on the evaluation of a model’s trustworthiness using their explanation system, and emphasize the importance of an oracle to assess the quality of explanations.

Sundararajan et al. (2017) bring a clear formalization on the problem of the attribution of deep network prediction to its input features. They emphasize the two conditions for a good attribution system, namely: a) sensitivity (i.e., if the input differs by 1 input component from the reference input, the attribution should be non-zero), and b) implementability invariance (i.e., if two architectures produce the same output for the same input, their attributions should be identical). They explain why state-of-the-art methods (especially gradients) do not meet these two criteria, and propose a method that complies – integrated gradient. Their method consists in integrating the gradient along component $i$ to get the attribution of input $i$.

Interpretability by attribution is a method that can be applied to any architecture, but as a consequence it does not consider the underlying mechanics for an architecture to explain the decision. A posteriori methods can also specialise in one type of architecture. In the particular case of the Transformer architecture (Vaswani et al., 2017) based on stacking self-attention layers, visualization tools like Bertviz (Vig, 2019) can be used to underpin an explanation of the model. Nonetheless, Brunner et al. (2019) warn on over-interpretation, when trying to explain self-attention. They especially argue that self-attention scores become a very complex mixture of interwoven words, while going deeper into the architecture, as a token is only responsible on average for 7.5% of the second-layer self-attention gradient. With this in mind, Chefer et al. (2020) do not restrict the influence of tokens to attention scores, but compute relevance and gradients back through the entire architecture, so as to compute the influence of each token on the decision.

The second approach considers interpretability by design. For example, Ming et al. (2019) provide an interpretable architecture for text classification that considers as implementation an RNN encoder; nevertheless, the model can consider any encoder. The idea for their architecture is to learn embeddings which well represent a special class of articles - i.e., "prototypes", and to cover as well as possible the latent space while relating to these articles. Our model builds on top of this architecture and additional details are provided in section 3.3.

3 Method

3.1 Corpora

Our aggregated corpus for training and evaluating our models consists of 1600 news articles from two collections of labeled articles that were obtained using two different approaches: Iorga et al. (2020) introduce a corpus of 1000 news articles on cybersecurity manually labeled by experts from news outlets, and Iorga et al. (2021) consider 600 more
articles that were extracted from selected Tweeter accounts. The distribution in terms of length is displayed in Figure 1; as it can be observed, more than half of the articles exceed the usual length of 512 tokens acceptable by most pretrained language models.

While accounting for the interpretability of our model, a more in-depth analysis of the articles from this aggregated dataset was required. An overwhelming majority of relevant articles disclose new vulnerabilities, either directly or through the description of an attack (or campaign). k-Means clustering was used to confirm this duality, while considering only relevant articles. Silhouette scores were computed for different number of clusters, which were afterwards visualized and cross-checked by hand to observe the split between attacks and explicit vulnerabilities. The optimal silhouette score was 0.15 for 2 clusters, followed by values lower than 0.13 for a higher number of clusters. As such, the duality is also confirmed while inspecting the most frequent tokens for each cluster (see Table 1).

| attack, user, researcher, vulnerability, device, attacker, security, malware, data, malicious |
| vulnerability, security, flaw, attacker, user, cve, update, code, version, windows |

Table 1: Keywords grouped by cluster.

A higher number of clusters would have put forward specific threats like password leaks, which represent a minority of articles and are often linked to attacks. Next, we need to consider the permissiveness of the classification: even when the technology compromised by the cyberthreat is very specific, the automated pipeline needs to label the corresponding articles as relevant. Moreover, the article needs to include specific details about the identified vulnerabilities, for example the corresponding attack vector. However, these details may represent a very limited part of the overall article.

While considering the subtle difference between the introduced topics in cybersecurity articles and the previous observations, the model needs to process the article as a whole, with corresponding inter-dependencies at word, sentence, or even paragraph levels. As such, bag-of-words approaches, such as Multinomial Naive Bayes (MNB, (Kibriya et al., 2004)), though easily interpretable by the user, will only make the most obvious decisions.

Besides the previous aggregated corpus, a second considerably larger and unlabeled corpus was also collected. The creation of this second dataset meets a need for creating an embedding space for the cybersecurity domain. For this purpose, recursive scraping was used on more than 20 news websites to collect webpages, which leads to a corpus of 65.8 million tokens and a vocabulary of 63 thousand words.

3.2 Data Pre-processing

Each article was pre-processed. First, accents, symbols, IP addresses, links, contractions, and residual dots were removed. Second, numbers were replaced by a # to report their presence. The dataset was then randomly split into a train set of 1000 articles and a test set of 600 articles using a stratification approach. It is important to emphasize the importance of recall, as we do not want to turn a "blind eye" on potential threats. Thus, the metric to be optimized for classification is F2-Score.

3.3 Interpretable Model

With the aim of designing an interpretable classifier, we combined the ProSeNet (Ming et al., 2019) architecture with Longformer (Beltagy et al., 2020) as an encoder for the entire articles.

ProSeNet considers a special layer, named the prototype layer, that takes the hidden state (in output of the encoder) as input, and its output is given to a standard classifier (dense layers and a sigmoid, in our case). The principle of the prototype layer is to compute the similarities (sim(x, y) = \frac{x^T y}{||x||^2 ||y||^2}) between the hidden state and learned vectors $p_1, ..., p_k$. These $k$ vectors (i.e., prototypes) are defined in the same latent space as the hidden state; as such, these vectors, modified by backpropagation during the epoch, are projected at the end
of epoch in the latent space of their nearest neighboring articles. If the dataset is large enough to cover the latent space, the training that includes this projection stabilizes, and the user ends up with a classification within which only the comparison between the input document and these prototypes matters for final prediction.

This construction of the decision is similar to human selection: when deciding, for example, if an article is relevant or not, it is natural to relate to similar, previously seen, articles. Further comparisons informs users that, if they want to make reasonable decisions, they should consider a wide range of articles. Thus, the number of prototypes being fixed, what an expert would expect from these prototypes is for each of them to be representative for various types of articles and to properly cover the entire possible semantic space of articles.

Nonetheless, the similarities computed with all prototypes have to be aggregated in making the final binary decision. This is where the method reaches its limits in terms of interpretability, as there is no telling how the classifier mixes the information of proximity to the prototypes in order to take its decision.

When looking at state-of-the-art work on automated text classification, the Transformer-based models stand out, pushing the boundaries of RNNs (LSTMs or GRUs cells) in terms of long-term dependencies encoding. Nonetheless, the quadratic-order computation of self-attention limits the number of tokens allowed as input. A frequently employed solution is to use only part of the text, or make several predictions (e.g., each centered on a paragraph) that are aggregated by means of voting for the final decision. However, the vulnerability in the article might end up being either completely neglected or erased for classification, if its description place in the article is limited. Therefore, the article should be considered as a whole to mitigate the risk of neglecting important details.

The Longformer (Beltagy et al., 2020) architecture overcomes the limits of classic Transformer models by changing the computation of self-attention. Instead of performing number of words operations, a window size is chosen, and relations are measured between the word and a sliding window of size around it. Beltagy et al. (2020) introduce three types of self-attention patterns distributed among the 12 encoding layers of the Longformer. The sliding window attention one has a dilation rate of 1 and it allows the efficient capturing of local information. This local information can then be aggregated thanks to dilated sliding-window self-attention patterns to capture more global information and to increase the receptive field. Global+sliding windows are mainly used in the final layers to provide task-specific tokens.

In our classification task, a single global self-attention is computed in the last encoding layer of the Longformer. The resulting architecture is presented Figure 2.

As the current implementations of ProSeNet are not adapted to our problem, we decided to adapt the implementation of Meyer (2019). The project is released as open-source and is available on Github.

### 3.4 Training Hyper-parameters

The Longformer has a window size of 128 tokens, whereas the other parameters are the default ones. The usual learning rate scheduler for Transformers was used for all training episodes, and a weighted binary cross-entropy was chosen to counterbalance the 2/3-1/3 irrelevant-relevant ratio. The classical Longformer model (i.e., without ProSeNet, with the usual classifier at the end) was trained both with and without pretraining. Pretraining is achieved on masked language modelling using the unlabelled cybersecurity dataset.

The interpretable architecture (Longformer+ProSeNet) relies on 15 prototypes. A first challenge was to make the training stable. The training of the entire architecture takes place as follows: we initialize the encoder with the weights of the previously trained Longformer as standalone, and we freeze it for the first epochs to stabilize training. Then, we unfreeze the layers and set a small learning rate for the last epochs. Four projections are computed during training.

### 3.5 On the Role of ProSeNet

Moreover, we were interested on the importance of the additional ProSeNet layer in terms of data separation between relevant and irrelevant articles. With this goal in mind, we scrutinized the influence of the different training steps (pretrained only, pretrained and finetuned) on the action of Longformer on data.

Nonetheless, when considering the distribution of embeddings in the latent space (see Figure 3),
only two groups emerge; reality is much more complex, as some irrelevant articles are not even linked to cybersecurity at all. This finer-grained characterization of the latent space is precisely what we expect ProSeNet to shed light upon. We fall in line with Ming et al. (2019) on the importance of diversity and sparsity for prototypes; however, it is also important to avoid having prototypes in the part of space where relevant and irrelevant articles coexist without a genuine separation or coherence, because of the lack of data. Prototypes thus need to represent a clearly relevant or clearly irrelevant part of the space. Indeed, they need to be representative for groups of articles, but if these groups are not clearly labeled as relevant or irrelevant, this will lead to a decrease in performance.

In terms of explainability, we emphasize the need for simplicity, as stated in the original study (Ming et al., 2019). Prototypes need to be represented briefly, without losing information required to identify them - as such, we simply present the articles by their title to the user.

4 Results

Table 2 reports the performance of various configurations, including a very simple interpretable baseline - a Multinomial Naive Bayes classifier trained using TF-IDF embeddings, while considering only the top 350 most relevant tokens chosen by feature importance.

In terms of the standard Longformer, the best performance was reached with pretraining, highlighting that additional cybersecurity context provides a significant boost in recall. Transformer-based models completely outperform MNB, particularly in terms of recall (98% for Transformer, only 62% for MNB), but at the heavy cost of interpretability. ProSeNet was designed to tackle this problem, but as we can see in Table 2, classification performance slightly deteriorates, with a 4% decrease in F2-Score when compared to the standard Longformer with pre-training.

5 Discussions

An encountered issue is that only 11 prototypes were actually selected because of duplicates (the links for the selected papers are presented in Table 3). These duplicates appeared during projection, despite adding a loss favoring the variety between prototypes to the weighted binary cross-entropy for training. To understand the emergence of duplicates, we have to remember that the amount of data might not be sufficient to cover well the entire latent space; thus, two prototypes in a poorly covered area are likely to be projected on the same article.

While taking a closer look at the articles, our first intuition (after having studied and tediously labelled the entries), is that the selected article seem to cover quite well the different cases. The balance between relevant and irrelevant articles is fulfilled. Articles 1, 2, 3, and 7 present general studies or miscellaneous events which are linked to cybersecurity, but not all to cyberthreat detection. Articles 4 and 8 study a cyberattack (without the disclosure of any vulnerability) and a legal case involving a cybercrime group; these are irrelevant. In contrast, relevant articles directly show vulnerability disclosure (articles 5, 9, and 10) or through cybercrime...
Figure 3: Projections with (right) and without (left) finetuning. Relevant articles are blue, irrelevant articles are pink.

Table 2: Comparison between different architectures.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F2-Score</th>
<th>Interpretable</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNB</td>
<td>0.84</td>
<td>0.92</td>
<td>0.62</td>
<td>0.66</td>
<td>Yes</td>
</tr>
<tr>
<td>Longformer (no pre-training)</td>
<td>0.87</td>
<td>0.76</td>
<td>0.95</td>
<td>0.90</td>
<td>No</td>
</tr>
<tr>
<td>Longformer (pre-trained)</td>
<td>0.86</td>
<td>0.73</td>
<td>0.98</td>
<td>0.92</td>
<td>No</td>
</tr>
<tr>
<td>Longformer+ProSeNet</td>
<td>0.87</td>
<td>0.78</td>
<td>0.91</td>
<td>0.88</td>
<td>Yes</td>
</tr>
</tbody>
</table>

(articles 6 and 11). The security experts responsible for the initial datasets also confirmed our intuition. Nonetheless, it is important to acknowledge that the evaluation of interpretability has not been achieved by a representative assembly of cybersecurity experts or members of SOCs, even though this is the minimum expected if we want organizations to trust our model. Still, the analysis provided by the experts grounded an encouraging overview in terms of sparsity, diversity, and simplicity (see 3.5).

In order to further support the degree to which the selected prototypes cover the latent space, Figure 4 adds the prototypes to the latent space. At first sight, their distribution does not reflect the previously mentioned diversity, as the articles from the region of indecision from the encoder are mostly not covered by prototypes. This further consolidates our previous argument, i.e. the lack of data, which causes prototypes to be obviously labeled articles, and thus hinders performance.

Moreover, affinity propagation was also been implemented, with the aim of finding a finer-grained split: 54 clusters are identified, which are way too many for only 600 articles. On closer inspection, the grouping was done according to the company affected by the vulnerability. An approach based on keyword mining for each article would be interesting to further explore.

6 Conclusions and Future Work

In this study we introduce a state-of-the-art architecture based on Longformer and ProSeNet to create an interpretable pipeline to automatically label emerging cybersecurity threats. The similarity with the human decision process, coupled with a balanced performance on rather small dataset (i.e., recall - the most important metric - reaches 91%, while precision is at 78%), argue for the model’s adequacy. Thus, in response to the initial research question, our architecture provides an efficient filter, while also ensuring interpretability.

The architecture is in place, but at a crossroads. First, further data collection is required in order to extend the training dataset. Once a substantial number of articles are collected, the quality of our pipeline will be re-assessed, while also including security experts to scrutinize the explanations of our model. This is a long-term endeavour in which
the experts would be asked to chose the closest prototype out of 3 candidates for a given article. Second, we aim to implement a finer-grained clustering of the articles and identify trending topics or sub-domains. In addition, we aim to include a customizable filter that will enable SOCs to select themes for relevant articles, as well as targeted applications, thus accounting for their deployed infrastructure.

Acknowledgments

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS – UEFISCDI, project number 2PTE/2020, YGGDRASIL – “Automated System for Early Detection of Cyber Security Vulnerabilities” and by the internal UPB program Proof of Concept.

References


Cross-lingual Offensive Language Identification for Low Resource Languages: The Case of Marathi

Saurabh Gaikwad\textsuperscript{1}, Tharindu Ranasinghe\textsuperscript{2}, Marcos Zampieri\textsuperscript{1}, Christopher M. Homan\textsuperscript{1}

\textsuperscript{1}Rochester Institute of Technology, USA
\textsuperscript{2}University of Wolverhampton, UK

T.D.RanasingheHettiarachchige@wlv.ac.uk

Abstract

The widespread presence of offensive language on social media motivated the development of systems capable of recognizing such content automatically. Apart from a few notable exceptions, most research on automatic offensive language identification has dealt with English. To address this shortcoming, we introduce \textit{MOLD}\textsuperscript{1}, the Marathi Offensive Language Dataset. \textit{MOLD} is the first dataset of its kind compiled for Marathi, thus opening a new domain for research in low-resource Indo-Aryan languages. We present results from several machine learning experiments on this dataset, including zero-shot and other transfer learning experiments on state-of-the-art cross-lingual transformers from existing data in Bengali, English, and Hindi.

1 Introduction

The presence of hate speech, cyber-bullying, and other forms of offensive language in online communities is a global phenomenon. Even though thousands of languages and dialects are widely used in social media, most studies on the automatic identification of such content consider English only, a language for which datasets and other resources such as pre-trained models exist (Rosenthal et al., 2021). In the past few years researchers have studied this problem on languages such as Arabic (Mubarak et al., 2021), French (Chiril et al., 2019), and Turkish (Çöltekin, 2020) to name a few. In doing so, they have created new datasets for each of these languages. Competitions such as OffensEval (Zampieri et al., 2020) and TRAC (Kumar et al., 2020a) provided multilingual datasets, which enabled the use of data augmentation methods (Ghadery and Moens, 2020), multilingual word embeddings (Pamungkas and Patti, 2019), and cross-lingual contextual word embeddings (Ranasinghe and Zampieri, 2020) to tackle this problem.

In this paper, we revisit the task of offensive language identification for low resource languages, focusing on Marathi, an Indo-Aryan language spoken by over 80 million people, most of whom live in India. Even though Marathi is spoken by a large population, it is relatively low-resourced compared to other languages spoken in the region. We collect and annotate the first Marathi offensive language identification dataset to date and we train a number of monolingual models on this dataset. Finally, we explore state-of-the-art cross-lingual learning methods to project predictions to Marathi from Bengali, Hindi, and English. We address two research questions in this paper:

RQ1: What is the impact of the dataset size in monolingual and cross-lingual models for offensive language identification? While the Marathi dataset is relatively small, cross-lingual transfer learning methods allow us to take advantage of larger available datasets in other languages.

RQ2: What is the influence of language similarity in cross-lingual predictions for offensive language identification? Previous work used English as the base language to make predictions in lower resourced languages. In this paper we use two Indo-Aryan languages, Bengali and Hindi, to project predictions into Marathi.

Our main contributions are the following:

1. We release \textit{MOLD}, the Marathi Offensive Language Dataset, with nearly 2,500 annotated tweets. \textit{MOLD} is the first dataset for offensive language identification in Marathi.

2. We evaluate the performance of several traditional machine learning models (e.g. SVMs)
and deep learning models (e.g. LSTM) trained on MOLD.

3. We apply cross-lingual transformers to offensive language identification in Marathi. We take advantage of existing data in English and in two Indo-Aryan languages, Hindi and Bengali, to project predictions to Marathi and we compare the results of these strategies. To the best of our knowledge, this is the first paper to study closely-related languages in transfer learning for offensive language identification.

4. In addition to MOLD, we make the code and the models freely available to the community.

2 Related Work

The problem of offensive content online has been widely studied using computational models. Researchers have trained system to recognize various types of such content such as cyberbullying, hate speech, and many others. In terms of computational approaches, early studies approached the problem using feature engineering and classical machine learning classifiers, most notably SVMs (Dadvar et al., 2013; Malmasi and Zampieri, 2017), while more recent work applied deep neural networks combined with word embeddings (Aroyehun and Gelbukh, 2018; Hettiarachchi and Ranasinghe, 2019). With the development of large pre-trained transformer models such as BERT and XLNET (Devlin et al., 2019; Yang et al., 2019), several studies have explored the use of general pre-trained transformers in offensive language identification (Liu et al., 2019; Ranasinghe et al., 2019; Bucur et al., 2021) as well retrained or fine-tuned models on offensive language corpora such as HateBERT (Caselli et al., 2020).

While the vast majority of studies address offensive language identification using English data (Yao et al., 2019; Ridenhour et al., 2020), several recent studies have created new datasets for various languages and applied computational models to identify such content in Arabic (Mubarak et al., 2021), Dutch (Tulkens et al., 2016), French (Chiril et al., 2019), German (Wiegand et al., 2018), Greek (Pitenis et al., 2020), Hindi (Bohra et al., 2018), Italian (Poletto et al., 2017), Portuguese (Fortuna et al., 2019), Slovene (Fišer et al., 2017), Spanish (Plaza-del Arco et al., 2021), and Turkish (Çöltekin, 2020). A recent trend is the use of pre-trained multilingual models such as XLM-R (Conneau et al., 2019) to leverage available English resources to make predictions in languages with less resources (Plaza-del Arco et al., 2021; Ranasinghe and Zampieri, 2020, 2021c,b; Sai and Sharma, 2021). This is made possible by the availability of the aforementioned datasets as well multilingual datasets made available at shared tasks such as HASOC 2019 (Mandl et al., 2019), TRAC 2018 and 2020 (Kumar et al., 2018, 2020a), and two tasks at SemEval: HatEval 2018 (Basile et al., 2019) and OffensEval 2020 (Zampieri et al., 2020).

3 Datasets

We present MOLD and four other datasets used in this work: the Bengali dataset (Bhattacharya et al., 2020) used in the TRAC-2 shared task (Kumar et al., 2020a)—henceforth BE, the Hindi dataset (Mandl et al., 2019) used in the HASOC 2019 shared task—henceforth HI, and the English datasets used in OffensEval, SemEval-2019 Task 6 and SemEval-2020 Task 12—henceforth EN-OLID (Zampieri et al., 2019) and EN-SOLID (Rosenthal et al., 2021), respectively.

To annotate MOLD, we followed OLID’s annotation scheme for English which has been replicated in SOLID and in datasets in Greek (Pitenis et al., 2020), Turkish (Çöltekin, 2020) and many other languages. OLID’s taxonomy comprises the following three levels:

- **Level A**: Offensive language identification: offensive (OFF) vs. non-offensive (NOT)
- **Level B**: Categorization of offensive language: targeted insult or thread vs. untargeted profanity
- **Level C**: Offensive language target identification: individual vs. group vs. other

This hierarchical taxonomy represents multiple types of offensive content in a single annotation scheme (e.g. targeted insults to an individual are often cyberbullying and targeted insults to a group are often hate speech) making it a great fit for cross-lingual learning applied to low-resource languages like Marathi. We used OLID level A labels to annotate MOLD and we map these labels to those included in the Bengali and Hindi datasets.

**MOLD** The Marathi dataset contains data collected from Twitter using the Twitter API. We aimed to achieve a similar distribution of offensive vs. non-offensive content present in OLID, which contains around 33% offensive and 67% non-offensive tweets. To make sure that both classes
were represented, we used both offensive and non-offensive keywords. For the offensive content we used 22 common curse words in Marathi and for the non-offensive content we used search phrases related to politics, entertainment, and sports along with the hashtag #Marathi.

We collected a total 2,547 tweets that were annotated by 6 volunteer annotators who are native speakers of Marathi with age between 20 and 25 years old and a bachelors degree. The annotation task is a binary classification, in which annotators assigned tweets as offensive (OFF) or not offensive (NOT). The annotators could flag a tweet as invalid if it contained four or more non-Marathi words. The final version of MOLD contains 2,499 annotated tweets randomly split 75%/25% into training and testing sets, respectively. We used Cohen’s kappa (Carletta, 1996) to measure agreement between pairs of annotators. We provided a common set of 100 instances to each of the three pairs of annotators and we report scores of 0.91 between A1 and A2, 0.79 between A3 and A4, and 0.77 between A5 and A6. Table 1 shows dataset statistics, including class distribution.

### Table 1: Number of instances and class distribution of NOT and OFF tweets in MOLD.

<table>
<thead>
<tr>
<th>Class</th>
<th>Training</th>
<th>Testing</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Offensive</td>
<td>1,205</td>
<td>418</td>
<td>1,623</td>
</tr>
<tr>
<td>Offensive</td>
<td>669</td>
<td>207</td>
<td>876</td>
</tr>
<tr>
<td>Total</td>
<td>1,874</td>
<td>625</td>
<td>2,499</td>
</tr>
</tbody>
</table>

Other Datasets: In addition to the Marathi dataset, we used the four aforementioned publicly available offensive language detection datasets presented in Table 2. OLID (EN-OLID) is one of the most popular offensive language datasets for English and we used its level A annotations (offensive vs. non-offensive) as labels. We used EN-SOLID, the largest available dataset of its kind as our second English dataset. EN-SOLID contains over nine million English tweets labeled in a weakly supervised manner (Rosenthal et al., 2021). EN-SOLID was created using an ensemble of four different models and provides, along with the class labels, the average and standard deviation of the confidence scores predicted by each model. We included only training examples with average confidence scores greater than 0.85 over all models, leaving us with 120,758 examples. Using both EN-OLID and EN-SOLID allows us to investigate the impact of training data size and help us answer RQ1.

To perform transfer learning from a closely-related language to Marathi, we used HI (Mandl et al., 2019). Both the English and Hindi datasets contain Twitter data making them in-domain with respect to MOLD. BE, the Bengali dataset (Bhattacharya et al., 2020), is different than the other datasets as it contain Facebook data and three classes, allowing us to compare the performance of cross-lingual embeddings on off-domain data but in a a language similar to Marathi. For Bengali we merged the classes overtly aggressive and covertly aggressive and map them to EN-OLID’s offensive class. Using both BE and HI in addition to the two English datasets allow us to investigate the impact of language similarity aiming to answer our RQ2.

### 4 Methods and Results

#### 4.1 Monolingual Models

We run several computational models on MOLD. We trained four classical machine learning classifiers, available in Scikit-learn (Pedregosa et al., 2011): Decision Trees, Naive Bayes, Random Forest, and SVM using bag of words (BoW), word unigrams, and word unigrams and bigrams combines using TF-IDF weighting. We took several pre-processing steps before extracting features such as removing numbers, extra spaces, special characters, and stop words.²

We implemented several deep learning models, such as multi layer perceptron (MLP), long short-term memory networks (LSTMs) with embedding layers, and bi-LSTMs with attention and word embedding layers. We used the Marathi word2vec embeddings released in Kumar et al. (2020b). We also experimented with several SOTA transformer models that support Marathi: multilingual BERT (BERT-m) (Devlin et al., 2019) and XLM-Roberta (XLM-R) (Conneau et al., 2019). XLM-R has an additional advantage: the embeddings are cross-lingual. This helps facilitate transfer learning across languages, as presented later in this section. We followed the same architecture described in Ranasinghe and Zampieri (2020) where a simple softmax layer is added to the top of the classification ([CLS]) token to predict the probability of a class label. For XLM-R, from the available two pre-trained models, we specifically used the XLM-R large model.

Table 2: Instances, sources, and labels in all datasets. F stands for Facebook and T for Twitter.

<table>
<thead>
<tr>
<th>Code</th>
<th>Language</th>
<th>Dataset</th>
<th>Instances</th>
<th>Source</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE</td>
<td>Bengali</td>
<td>TRAC</td>
<td>4,000</td>
<td>F</td>
<td>overtly aggressive, covertly aggressive, non aggressive</td>
</tr>
<tr>
<td>EN-OLID</td>
<td>English</td>
<td>OLID</td>
<td>14,100</td>
<td>T</td>
<td>offensive, non-offensive</td>
</tr>
<tr>
<td>EN-SOLID</td>
<td>English</td>
<td>SOLID</td>
<td>120,758</td>
<td>T</td>
<td>offensive, non-offensive</td>
</tr>
<tr>
<td>HI</td>
<td>Hindi</td>
<td>HASOC</td>
<td>8,000</td>
<td>T</td>
<td>hate offensive, non-hate-offensive</td>
</tr>
</tbody>
</table>

For both classical and deep learning models we finetuned hyperparameters manually to obtain the best results for the validation set created using a 0.8:0.2 split on the training data. As the deep learning models tend to overfit, we evaluated the model on the validation set once in every 100 training batches. We performed early stopping if the validation loss did not improve over 10 evaluation steps. All the deep learning experiments were run on an Nvidia Tesla K80 GPU.

For classical models, we used the baseline architectures described in Section 3.1 and finetuned the hyperparameters to obtain the best results for the validation set created using a 0.8:0.2 split on the training data. As the deep learning models tend to overfit, we evaluated the model on the validation set once in every 100 training batches. We performed early stopping if the validation loss did not improve over 10 evaluation steps. All the deep learning experiments were run on an Nvidia Tesla K80 GPU.

Table 3 shows the results obtained by all monolingual models on MOLD’s test set in terms of both Macro F1 and Weighted F1. We use both metrics due to the data imbalance in MOLD. With the exception of MLP, all of the deep learning models outperformed the classical ones. This is somewhat surprising as classical models tend to outperform deep models on relatively small datasets like MOLD, but it corroborates the findings from recent competitions on this topic (Basile et al., 2019). Of the deep learning models, XLM-R transformers provided the best results with a 0.91 macro F1 score.

<table>
<thead>
<tr>
<th>Features</th>
<th>Model</th>
<th>M F1</th>
<th>W F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embeddings</td>
<td>XLM-R</td>
<td>0.9103</td>
<td>0.9210</td>
</tr>
<tr>
<td>Embeddings</td>
<td>BERT-m</td>
<td>0.8852</td>
<td>0.8994</td>
</tr>
<tr>
<td>Embeddings</td>
<td>LSTM</td>
<td>0.8400</td>
<td>0.8409</td>
</tr>
<tr>
<td>Embeddings</td>
<td>Bi-LSTM</td>
<td>0.8238</td>
<td>0.8251</td>
</tr>
<tr>
<td>BoW</td>
<td>Random Forest</td>
<td>0.7686</td>
<td>0.7796</td>
</tr>
<tr>
<td>Embeddings</td>
<td>MLP</td>
<td>0.7541</td>
<td>0.7830</td>
</tr>
<tr>
<td>BoW</td>
<td>SVM</td>
<td>0.7489</td>
<td>0.7813</td>
</tr>
<tr>
<td>BoW</td>
<td>Naive Bayes</td>
<td>0.7223</td>
<td>0.7597</td>
</tr>
<tr>
<td>BoW</td>
<td>Decision Tree</td>
<td>0.7028</td>
<td>0.7395</td>
</tr>
</tbody>
</table>

Table 3: Monolingual results for Marathi ordered by macro (M) F1. We also report weighted (W) F1 scores.

4.2 Cross-lingual Models

The main appeal of transfer learning is its potential to leverage models trained on data from outside the domain of interest. This can be particularly helpful for boosting the performance of learning on low-resource languages like Marathi. The recent success of XLM-R cross-lingual transformers with transfer learning in offensive language identification for low resource languages (Ranasinghe and Zampieri, 2020) confirms that this is a feasible approach. In these experiments, however, the transfer learning’s base language was English whereas here we use two languages related to Marathi: Bengali and Hindi, in order to evaluate the extent to which language similarity boosts transfer learning performance.

Transfer Learning We first trained the XLM-R model separately on the BE, HI, EN-OLID and EN-SOLID datasets. Then we saved the weights of the transformer model and the softmax layer and used these weights to initialize the weights of the transformer-based classification model for Marathi. TL row in Table 4 shows the results obtained by the cross lingual models with XLM-R. The use of transfer learning substantially improved the monolingual results. With 8,000 and 4,000 training instances, respectively, the transfer learning model achieved macro F1 scores of 0.9401 from Hindi and of 0.9345 from Bengali, respec-
tively, outperforming the results obtained using the two English datasets, EN-OLID and, especially, EN-SOLID, each contain more instances than either the Hindi or the Bengali dataset, yet they fail to outperform either as the base dataset in our transfer learning experiments, suggesting that language similarity played a positive role in transfer learning.

**Zero shot learning** To further observe the impact of language similarity in transfer learning, we performed Zero shot learning, where the XLM-R model was trained on the other datasets and tested on the Marathi test set. According to the results in Zero-shot row of Table 4 HI outperforms all the other languages in Zero shot too.

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Dataset</th>
<th>M F1</th>
<th>W F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer Learning</td>
<td>HI</td>
<td>0.9401</td>
<td>0.9492</td>
</tr>
<tr>
<td></td>
<td>BE</td>
<td>0.9345</td>
<td>0.9422</td>
</tr>
<tr>
<td></td>
<td>EN-SOLID</td>
<td>0.9321</td>
<td>0.9399</td>
</tr>
<tr>
<td></td>
<td>EN-OLID</td>
<td>0.9298</td>
<td>0.9385</td>
</tr>
<tr>
<td>Zero-Shot</td>
<td>HI</td>
<td>0.8396</td>
<td>0.8461</td>
</tr>
<tr>
<td></td>
<td>BE</td>
<td>0.8115</td>
<td>0.8176</td>
</tr>
<tr>
<td></td>
<td>EN-SOLID</td>
<td>0.7954</td>
<td>0.8004</td>
</tr>
<tr>
<td></td>
<td>EN-OLID</td>
<td>0.7854</td>
<td>0.7901</td>
</tr>
</tbody>
</table>

Table 4: Transfer learning results ordered by macro (M) F1 for Marathi. We also report weighted (W) F1 scores.

**Few shot learning** Finally, we evaluated each of the languages performance in few shot learning with Marathi. We retrained offensive language identification XLM-R models from other languages on 100, 200, 300 etc. instances from Marathi. As shown in Figure 2 HI tops other languages in all the few shot experiments making it further clear that transfer learning from a more similar language is effective in offensive language identification.

5 Conclusion and Future Work

This paper introduced MOLD, the first offensive language dataset for Marathi. We evaluated the performance of several machine learning models trained to identify offensive content in Marathi. Our results show that applying cross-lingual contextual word embeddings substantially improved performance over monolingual models. Furthermore, we showed that XLM-R with transfer learning from Hindi outperforms all of the other methods we tested. The results obtained by our models confirm that closely related languages provide an advantage in our transfer learning experiments, answering our RQ2. This is likely due to the fact that Hindi and Marathi are typologically related and also because these languages are in a situation of language contact sharing cultural background.

To the best of our knowledge, this paper is the first to address the question of language similarity in cross-lingual learning for offensive language identification. With respect to our RQ1, our results show that the difference in performance between transfer learning strategies from OLID and from SOLID is minimal. SOLID is more than eight times larger than OLID, suggesting that beyond a certain point, more instances do not necessarily yield significant performance improvements in transfer learning. Finally, we believe that the findings presented in this paper can open a wide range of avenues to offensive language identification applied to other low resource languages, particularly from the Indo-Aryan family.

**MOLD** is the official dataset for Marathi at the HASOC 2021\(^3\) shared task on Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages. We are expanding the annotation of this dataset to the levels B and C of OLID’s annotation taxonomy. This will provide us with the opportunity to test computational models to identify the type and target of offensive posts in Marathi. As future work, we would like to evaluate the performance of transfer learning from Dravidian languages spoken in India such as Tamil and Telugu to analyze the interplay between language similarity and cultural overlap in cross-lingual offensive language identification as in Ranasinghe and Zampieri (2021a).

---

\(^3\)https://hasocfire.github.io/hasoc/2021/index.html
Acknowledgment

We would like to thank the dataset annotators who helped us with the annotation of MOLD. We further thank the anonymous RANLP reviewers for the insightful feedback provided.

References


Saurav Kumar, Saunack Kumar, Diptesh Kanojia, and Pushpak Bhattacharyya. 2020b. “a passage to India”: Pre-trained word embeddings for Indian languages. In *Proceedings of SLTU*.


447


Relying on Discourse Analysis to Answer Complex Questions by Neural Machine Reading Comprehension

Boris Galitsky
Oracle Inc. / Redwood Shores, CA, USA
boris.galitsky@oracle.com

Dmitry Ilvovsky
National Research University Higher School of Economics / Moscow, Russia
dilvovsky@hse.ru

Elizaveta Goncharova
National Research University Higher School of Economics / Moscow, Russia
egoncharova@hse.ru

Abstract

Machine reading comprehension (MRC) is one of the most challenging tasks in natural language processing domain. Recent state-of-the-art results for MRC have been achieved with the pre-trained language models, such as BERT and its modifications. Despite the high performance of these models, they still suffer from the inability to retrieve correct answers from the detailed and lengthy passages. In this work, we introduce a novel scheme for incorporating the discourse structure of the text into a self-attention network, and, thus, enrich the embedding obtained from the standard BERT encoder with the additional linguistic knowledge. We also investigate the influence of different types of linguistic information on the model’s ability to answer complex questions that require deep understanding of the whole text. Experiments performed on the SQuAD benchmark and more complex question answering datasets have shown that linguistic enhancing boosts the performance of the standard BERT model significantly.

1 Introduction

Machine reading comprehension (MRC) reflects the ability to read and understand an unstructured text and answer questions regarding it. Aiming to find the relevant answer to a question in the form of a text span, the MRC models should demonstrate deep understanding of the language and text organization.

Transformer models that achieve state-of-the-art results on multiple natural language processing (NLP) tasks have been successfully applied to the MRC. However, while the ideal MRC model should read most words superficially and pay attention only to the essential ones (Wang et al., 2017), the attention mechanism in the standard transformers attends to all words without explicit constraint which results in inaccurate concentration on some less important text spans. Lately, the researchers have actively examined the ability of the deep learning (DL) models to understand language and build accurate linguistic-enhanced internal representation.

Recent works have revealed that traditional DL models that ignore additional linguistic knowledge, such as syntax or semantic, achieve lower accuracy on such complex tasks as natural language understanding (NLU) or MRC (Roth and Lapata, 2016; Marcheggiani and Titov, 2017; He et al., 2018). It has been shown that incorporating explicit syntactic (Hu et al., 2019) and semantic (Zhang et al., 2020a) relations into the attention mechanism leads to better linguistically motivated word representations beneficial for the MRC task.

Moreover, providing exact, concise answers frequently requires not just syntactic/meaning similarity but an overall structure of thoughts expressed by an author (Galitsky et al., 2013), i.e., some claims introduced by an author and logical connections existing among them. This information is encoded by discourse structure of a text that, as long as syntax and semantic, is believed to provide valuable information that could help the model to capture all the hidden dependencies existing in the text and to pay attention only to the relevant words while answering the corresponding question.

In this paper, we explore if and how discourse-level features (discourse relations connecting the text spans), fed to a neural MRC model on top of syntactic and semantic features or independently, can help to answer complex, long, multi-sentence questions. We intend to develop a neural method
that selects relevant words by only considering the related subset of words, w.r.t. syntactic, semantic, and discourse-level importance. To provide feature encoding we use a self-attention network (SAN) enriched with the discourse features (such as explanation, condition, etc.) retrieved from a text and combine it with the classical transformer encoder to build linguistically-enhanced text representation.

Overall, the contribution of this paper is three-fold: first, we introduce a novel discourse-aware transformer-based model to construct the enriched internal representation of the text. Second, we develop an ensemble MRC model that combines syntax, semantic, and discourse MRC components. Third, we conduct experiments on various question-answering (QA) datasets to assess the ability of the linguistically enriched model to answer complex questions and estimate the influence of each source of linguistic information.

2 Related Work and Background

2.1 Machine Reading Comprehension

Span-based MRC, which is the main focus of this work, is quite a challenging task, as we expect the model not only to identify the relevant document that contains a possible answer but to retrieve the exact text fragment that answers the question. There has been a lot of studies on solving this task with attentive models (Kadlec et al., 2016; Yuan et al., 2018; Guo et al., 2019).

Recently, the pre-trained contextual language models (LMs) such as ELMO (Peters et al., 2018), BERT (Devlin et al., 2019), or a series of GPTs (Radford et al., 2018) have shown state-of-the-art results on the number of NLU benchmarks which has attracted the researchers’ interest toward utilizing these models for MRC. Despite the increasing popularity of these LMs, several studies have revealed that textual representation provided by them relies purely on the context of each word and, generally, neither the syntactical nor semantic organization of the text is considered. As this information is crucial for MRC, the novel techniques to incorporate syntactic and semantic knowledge into the pre-trained LMs have been the main focus of the latest works.

2.1.1 Syntactic-aware Models

Recent attempts to turn neural network algorithms into more structure-aware ones have discovered the incorporation of external memories in the context of recurrent neural networks. The idea is to use multiple memory slots outside the recurrence to piece-wise store representations of the input. Read and write operations for each slot can be modeled as an attention mechanism with a recurrent controller. Cheng et al. (2016), for example, leverage memory and attention to empowering a recurrent network with stronger memorization capability and more importantly the ability to discover relations among tokens. This is realized by inserting a memory network module in the update of a recurrent network together with attention for memory addressing. The attention acts as a weak inductive module discovering relations between input tokens and is trained without direct supervision. The experiments performed on NLI datasets showed that the superiority of the modified model over the vanilla LSTMs.

In more recent work (Zhang et al., 2020b), the authors benefit from the performance of the BERT model on span-based MRC tasks and sponsor it with the syntax-guided SAN. They design an informative method that can selectively pick out important words by only considering the related subset of syntactically important context inside each input sentence explicitly. With the guidance of syntactic structure clues, the syntax-guided method could give more accurate attentive signals and reduce the impact of the noise brought about by lengthy sentences. The authors extend the self-attention mechanism with syntax-guided constraint, to capture syntax-related parts with each concerned word. Specifically, they adopt a pre-trained dependency syntactic parse tree structure to produce the related nodes for each word in a sentence, namely syntactic dependency of interest, by regarding each word as a child node, and the syntactic dependency of interest consists of all its ancestor nodes and itself in the dependency parsing tree. The syntax encapsulating into the model should provide a better understanding of the long or unanswerable questions, which is a big obstacle for the existing MRC models.

2.1.2 Semantic-aware Models

Frequently, DL models suffer from insufficient contextual semantic representation and learning. So, the way of constructing semantic-aware LMs has also attracted wide attention in research.

To provide contextual semantic representation to the DL models, Strubell et al. (2020) propose linguistically-informed self-attention (LISA), which is used for the semantic role labeling (SRL) task. The model proposed by the authors is end-to-end, and it is trained to predict part of speech
tags, provide parsing, attend to syntactic parse dependents, and, finally, assign semantic role labels to the model. This architecture has been applied to enlarge the contextual representation provided by BERT with the additional semantic information.

In (Zhang et al., 2020a), the authors propose to use SRL task to integrate the text representation provided by BERT with the contextual explicit semantic embedding, the introduced model is called \textit{Sem-BERT}. Sem-BERT is intended to handle multiple sequence inputs, the words in the input sequence are passed to semantic role labeling to obtain multiple predicate-derived structures to form a semantic embedding. In parallel, the input sequence is segmented to subwords (if any) by BERT word-piece tokenizer, then the subword representation is transformed back to word-level via a convolutional layer to obtain the contextual word representations. Finally, the word representations $H$ and semantic embedding $H_{\text{sem}}'$ are concatenated to form the joint representation.

Despite there is a number of works encapsulating the syntactic and semantic information about the text into the DL models, there is still a lack of research that considers discourse organization, which also introduces relevant linguistic knowledge essential for MRC, and other downstream challenges. In this work, we propose a way to encode the discourse structure of the text by neural network and enrich the text embeddings constructed by BERT with this information. Then, we aim to assess the influence of discourse, semantic, and syntactic features on the MRC task.

\subsection{Discourse Structure}

In this section, we introduce the definition of discourse structure that we propose to integrate into the MRC model. Any coherent text is structured so that we can derive and interpret the information. This structure shows how discourse units (text spans such as sentences or clauses) are connected and relate to each other. Discourse analysis reveals this structure and describes the relations that hold between text units in the document. Several theories have been proposed in the past to describe the discourse structure, among which the Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) is one of the most popular. RST divides a text into minimal atomic units, called Elementary Discourse Units (EDUs). It then forms a tree representation of discourse called a Discourse Tree (DT) using rhetorical relations (Elaboration, Explanation, etc.) as edges, and EDUs as leaves. EDUs linked by a rhetorical relation are also distinguished based on their relative importance in conveying the author’s message: the nucleus is the central part, whereas the satellite is the peripheral part. Nucleus units consist of basic information and satellite units contain additional information about the nucleus.

An exploration of coherence relations in frameworks such as RST has experienced a revival in the decade in English and a few other languages (Matthiessen and Teruya, 2015; Maziero et al., 2015; Zeldes, 2016) which has led to a grown number of applications of discourse analysis. For example, discourse parsers are used in argumentation mining in online discussions, summarization, QA systems, and machine translation (Benamara et al., 2017; Durrett et al., 2016; Peldszus and Stede, 2016; Chakrabarty et al., 2020). We claim that incorporating this additional discourse information provided by state-of-the-art parsers could be beneficial for DL models performing MRC. We are motivated to improve the self-attention layer appended to the top of the transformer encoder to enrich the contextualized word representation with information from its neighbors and the relations from the dependency parse trees.

\section{MRC System Extended with Discourse Relations}

In this paper, we present the novel discourse-aware attentive model designed to perform the MRC task. Our approach is inspired by syntax-guided BERT (Zhang et al., 2020b), while instead of encapsulating the syntactic dependencies among the words, we pre-process the discourse parse tree and observe the EDUs as long as the specific discourse relations connecting them.

We introduce the architecture of the transformer-based encoder empowered with the discourse knowledge about the input text in Section 3.1. As we aim to assess the influence of all three types of linguistic information, in Section 3.2, we present the final MRC system designed as the ensemble of state-of-the-art syntactic, semantic, and the proposed discourse-aware attention components.

\subsection{Discourse-aware Model}

In this section, we describe a method for incorporating discourse relations into the transformer-based model explicitly. As well as the syntactic
dependency parse tree, the discourse structure can be represented as a hierarchically organized tree, where the leaves are the text spans and the edges denote the type of relations connecting them. Thus, we propose to modify the SAN appended to the top of vanilla transformer-based encoder to make it able to process the discourse text organization, and, thus, to utilize this additional linguistic feature for MRC.

3.1.1 Discourse-aware Self-attention Layer

Our discourse-aware language model is trained to provide the vector representation of the text enriched by the discourse relations connecting text units. To obtain this representation we use the standard transformer encoder to calculate contextual representation of the text, then the obtained vector is passed through the discourse-aware SAN, which is designed to encapsulate the discourse structure into the embedding of the sequence. Finally, the discourse-aware representation is aggregated with the output of the pure transformer, this final embedding goes through the task-specific layer to perform the MRC task. The overall model architecture is presented in Fig. 1.

Generally, the main difference between the discourse-aware language model and the traditional transformer-based model is as follows. In traditional transformers, the word attends to both sides of the context, while in the discourse-aware model we would like each word to attend to its discourse-dependent ancestors. This forces a multi-head attention mechanism to analyze the dependency among tokens w.r.t. the rhetoric relations connecting them. As we have already mentioned, the discourse structure of the text is represented by the DT. In this section, we will present the approach for incorporating this DT into the SAN.

To provide the discourse structure of the text we use a state-of-the-art discourse parser (Joty et al., 2013) which constructs a hierarchically organized dependency tree for the input text. The text annotated with discourse relations will be transmitted to the attention network. Whereas the SAN cannot encapsulate the whole discourse tree, we need to detect the essential dependencies existing among words that should be included in the model. Each discourse unit (sequence of words) corresponding to some leaf in DT is connected to its ancestor non-terminal node labeled by the rhetoric relation referred to this EDU. For example, in the passage and its DT shown in Fig. 3, the words do not attend to all of the left and right neighbors in the context, on the contrary, the words finds and clinicians are connected to their ancestor labeled by Attributions, while the pneumonia attends to its ancestor Cause which also depends on Attributions. This sequence of connections fully reflects the discourse organization of the text.

Formally, given input token sequence $S = \{s_1, s_2, ..., s_n\}$ of length $n$, we first pass it through the discourse parser to split into the EDUs and generate the discourse dependencies existing between them. The input sequence after parsing is enriched with the discourse relations $S_{rel} = \{rel_1, edu_1, rel_2, edu_2, ..., rel_m, edu_m\}$ of length $m$ where $edu_i = \{s_k, s_{k+1}, ..., s_{k+K}\}$, and $K$ is the number of tokens assigned to the $i$th EDU. We should notice that $edu_i$ could be an empty set if $rel_i$ connects two non-terminals nodes corresponding to the sub-trees in the DT (see contrast-elaboration relations in Fig. 3). Then, we should retrieve the ancestor nodes for each of the word $s_i$ and the rhetoric relation $rel_i$. To provide this we traverse the discourse dependency tree, and the ancestor node set $P_i$ is derived for each $s_i$ and $rel_i$. Finally, in an analogy with syntax-guided SAN a discourse dependency of interest mask $M$ is obtained. $M$ is $(n + m) \times (n + m)$ matrix, where the elements in each row denote the dependency mask of all tokens to the row-index token. $M[i, j] = 1$ means that token $s_i$ is the ancestor node of token $s_j$.

$$M[i, j] = \begin{cases} 1 & \text{if } j \in P_i \text{ or } j = i \\ 0 & \text{otherwise.} \end{cases}$$

To obtain the discourse-aware representation of the text we project the last layer output $H$ of size $L$ calculated by the original transformer encoder into the distinct key, value, and query representations of dimensions $<L \times d_k, L \times d_q, L \times d_v>$, respectively, denoted $<K'_i, Q'_i, V'_i>$ for each headword $i$. Then a dot product is computed to score key-query pairs with the dependency of interest mask to obtain attention weights of dimension $L \times L$, denoted $A'_i$:

$$A'_i = \text{Softmax}\left(\frac{M \cdot (Q'_iK'_iT)}{\sqrt{d_k}}\right)$$

The attention weight $A'_i$ is multiplied by $V'_i$ to obtain the discourse-aware token representations: $W'_i = A'_iV'_i$. $W'_i$ for all heads are concatenated and passed through a feed-forward layer. After passing through another feed-forward layer, a layer
normalization is applied to the sum of output and initial representation to obtain the final $H'_{\text{disc}} = \{h'_0, h'_1, ..., h'_n\}$.

Finally, we summarize the two text representations, where the former is obtained from the standard transformer encoder $H$, and the latter is the discourse-aware text representation $H'_{\text{disc}}$, finally $H_{\text{disc}} = H + H'_{\text{disc}}$.

### 3.1.2 Answer Detection

Having identified the model for calculating discourse-aware text representation, we could proceed with the MRC task. MRC is the ability to answer the question based on the input paragraph of the text. As we have already mentioned, in this work, we consider a so-called span-based MRC, where the answer should be found as the span of the input passage referring to the question. Formally, we can define the span-based MRC by a triple $< P, Q, A >$, where $P$ is the text paragraph which is the basis for the question $Q$, and $A$ is the correct answer to the question.

The input data which is fed to the transformer encoder is performed as $[\text{CLS}] P [\text{SEP}] Q [\text{SEP}]$, where the $[\text{CLS}]$ and $[\text{SEP}]$ are the special tokens utilized in the BERT model.

We use BERT model as the transformer encoder, so, the $[\text{CLS}]$ token representation calculated by the BERT encoder for the input sequence is used as the contextualized representation $H$ of the whole text passage and question. Finally, $H$ goes through the linguistically enriched SAN in order to obtain $H_{\text{ling}}$ and $H_{\text{ling}}$, where $\text{ling} \in \text{[synt, sem, disc]}$ that refers to the syntax, semantic, and discourse-aware SAN, respectively. $H_{\text{ling}}$ is fed to a linear layer to obtain the probability distribution over the start and end positions of the answer in the text through a softmax layer.

In the work, we propose to analyze the influence of various linguistic characteristics on the MRC. So, for the experiments, we will use both the standalone $H_{\text{ling}}$, and their combination, calculated as the sum of the individual $H_{\text{ling}}$.

### 3.2 MRC Pipeline

Fig. 2 demonstrates the architecture of the whole pipeline that we introduce to perform the MRC task. All in all the main components of the model are as follows:

1. Linguistic data preparation, which extracts, organizes, and aligns linguistic features at various levels of knowledge abstraction. There the system parses the input text passage to obtain relevant linguistically enriched structures that will be further utilized in neural model performance. Discourse after-parser is responsible for enrichment of the input sequence with the discourse relations revealed form the DT, $S_{\text{rel}}$.

2. Deep learning component actually performing MRC. This block provides encoding of the input passage and related questions using the classical transformer encoder as long as the additional linguistic feature extraction. The output of the context aggregation block is the representation $H_{\text{ling}}$, which is the sum of context-based text embedding and the embedding provided by linguistically-guided SAN.
Finally, this $\overline{H}_{ling}$ is used to perform the MRC task.

The discourse SAN block is the one introduced in Section 3.1.1. To provide the syntactic- and semantic-aware models we use the state-of-the-art models described in Sections 2.1.1 and 2.1.2. Specifically, we implement syntax-guided SAN by (Zhang et al., 2020b) and Sem-Bert by (Zhang et al., 2020a). These models are able to encapsulate the corresponding linguistic features into the transformer-based models that help to achieve an accuracy gain for the tasks related to MRC.

4 Experiments

In this work, we rely on four QA datasets with long, complex, multi-hop questions to observe if/how syntactic, semantic, and, mainly, discourse-level features help to provide the correct answers. As the baseline, we use fine-tuned BERT model. Besides, we compare the performance of our system with the current state-of-the-art results published or obtained from the leaderboard for the corresponding dataset.

4.1 Datasets and Setup

The experimental evaluation has been performed on several extracting reading comprehension English datasets. First, we verified the model on the well-known SQuAD datasets (Rajpurkar et al., 2016, 2018). then we evaluated how the introduced MRC model can cope with the more complex questions that require language comprehension and understanding of the full text rather than just a small paragraph. As the example of complex questions datasets, we consider NewsQA (Trischler et al., 2017), QA in Context (QuAC) dataset (Choi et al., 2020), and multi-sentence questions (MSQ) (Burchell et al., 2020).

Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions posed by crowd-workers on a set of Wikipedia articles, where the answer to every question is a segment of text from the corresponding reading passage. SQuAD contains more than 100,000 question-answer pairs on 500 articles, which is significantly larger than previous reading comprehension datasets. We use two versions of this corpus: SQuAD 1.1 and SQuAD 2.0, where the latter also includes unanswerable questions so that we can test the ability of the model to detect the questions that cannot be answered based on the provided paragraph. F1 score that measures the weighted average of the word-level precision and recall rate is used to evaluate the performance of the models.

NewsQA dataset consists of 100K QA pairs written by humans for CNN news articles. Answers are typically the multiword spans of the source text, as in the SQuAD there are unanswerable questions.
presented. The main challenge of this dataset is that a significant proportion of questions cannot be solved without reasoning, i.e. understanding conceptual overlap or identifying the synonyms.

**MSQ dataset** uses the Stack Exchange Data Explorer, an open-source tool for running arbitrary queries against public data from the Stack Exchange network. The authors of this corpora chose 93 sites within the network and queried each site for entries with at least two question marks in the body of the question. Also, the authors filtered too short (under 5 characters) and too long (over 300 characters), and badly formed questions. After cleaning and processing, 162,745 questions from 93 topics were extracted. This dataset includes the questions that consist of several sequential questions, and in order to answer them right, they should be considered as the one. We are not aware of any works that has attempted to improve QA performance on MSQs so far.

**QuAC dataset** has 100K QA pairs created by two crowd workers who are asking and answering questions about a hidden Wikipedia text. This dataset is aimed at enabling the MRC model to answer the latest question by comprehending not only the given context passage but all the dialogue that has been seen so far.

### 4.2 Results

To assess the influence of different linguistic features on the model performance we divided our experiments into two parts. Firstly, we provide the results on SQuAD datasets, then we present the evaluation on the more complex (w.r.t. the questions' design) NewsQA, QuAC, and MSQ datasets. In all experiments, we calculate F1 score as the weighted average between precision and recall. The results achieved by the introduced MRC model are presented in the bottom block of the table. We also show the results of the state-of-the-art models presented in the literature or public leaderboards (* symbol is used to refer to the unpublished works) for the available datasets (upper block). The results achieved by the MRC models relying on discourse information are in bold.

**SQuAD.** Our performance on both SQuAD 1.1 and 2.0 test data is shown in Table 1. The default MRC (baseline) employs neither syntactic nor semantic information, this is a typical fine-tuned cased BERT used as the encoder for the question and the passage. As we move towards syntactic, semantic, and discourse levels the average performance gain is 2.2, 3.4, and 3% respectively. The improvement of the integrated system is 5.4%. Despite the fact that the introduced model could not outperform the best both single (such as ALBERT) and ensemble (FPNet) models, we can observe that it boosts the default linguistic-free baseline essentially.

**Complex datasets.** Table 2 shows the result on NewsQA, QuAC, and MSQ. As we proceed to-

<table>
<thead>
<tr>
<th>Dataset/settings</th>
<th>v1.1 test</th>
<th>v2.0 test</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SQuAD leaderboard</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPNet*</td>
<td>-</td>
<td>93.18</td>
</tr>
<tr>
<td>Retro-Reader (Zhang et al., 2020c)</td>
<td>-</td>
<td>92.98</td>
</tr>
<tr>
<td>ALBERT (Lan et al., 2020)</td>
<td>-</td>
<td>92.20</td>
</tr>
<tr>
<td>LUKE*</td>
<td>95.4</td>
<td>-</td>
</tr>
<tr>
<td>Baseline</td>
<td>88.61</td>
<td>83.98</td>
</tr>
<tr>
<td>Syntax MRC</td>
<td>89.90</td>
<td>87.13</td>
</tr>
<tr>
<td>Semantic MRC</td>
<td>90.60</td>
<td>88.76</td>
</tr>
<tr>
<td>Discourse MRC</td>
<td>90.08</td>
<td>88.60</td>
</tr>
<tr>
<td>Syntax w. semantic w. discourse MRC</td>
<td>93.14</td>
<td>90.20</td>
</tr>
</tbody>
</table>

**Table 1:** F1 scores (%) on SQuAD 1.1 (v1.1) and SQuAD 2.0 (v2.0) datasets.

<table>
<thead>
<tr>
<th>Dataset/settings</th>
<th>NewsQA</th>
<th>QuAC</th>
<th>MSQ</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>literature + QuAC leaderboard</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpanBERT (Joshi et al., 2020)</td>
<td>73.6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DecaProp (Tay et al., 2018)</td>
<td>66.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RoR*</td>
<td>-</td>
<td>74.9</td>
<td>-</td>
</tr>
<tr>
<td>FlowQA (Huang et al., 2019)</td>
<td>-</td>
<td>64.1</td>
<td>-</td>
</tr>
<tr>
<td>Baseline</td>
<td>66.48</td>
<td>65.69</td>
<td>60.66</td>
</tr>
<tr>
<td>Syntax MRC</td>
<td>70.95</td>
<td>71.09</td>
<td>66.79</td>
</tr>
<tr>
<td>Semantic MRC</td>
<td>71.84</td>
<td>70.15</td>
<td>66.55</td>
</tr>
<tr>
<td>Discourse MRC</td>
<td>72.13</td>
<td>72.40</td>
<td>67.80</td>
</tr>
<tr>
<td>Syntax w. semantic w. discourse MRC</td>
<td>75.05</td>
<td>74.88</td>
<td>71.65</td>
</tr>
</tbody>
</table>

**Table 2:** F1 scores (%) on complex questions datasets. The performance of other MRC models on MSQ dataset has not been published yet.
wards evaluation in the datasets of more complex questions, the performance drops up to 20%. Analogously to Table 1, the default MRC employs none of the additional linguistic information. Whereas the absolute performance value is lower than in Table 1, the performance boost due to linguistic information is higher. The average contributions of syntactic, semantic, and discourse levels are 5.3, 5.2, and 6.5% respectively. One can observe that contribution of discourse-level features is the highest in this evaluation domain of longer, multi-sentence questions (MSQ). The improvement of the integrated system is almost 11% for MSQ, and 9.5% on average. Hence, the more long and complex the questions are, the higher the impact of linguistic information, especially discourse-level. We should also mention that the introduced ensemble model outperforms both the stand-alone fine-tuned BERT and current state-of-the-art models for NewsQA and achieves comparable results on QuAC.

4.3 Case Study

Finally, let us consider a case study, where the linguistic-free BERT model provides the wrong result answering the question, while the introduced discourse-aware MRC model can answer the question correctly. Bellow, there are an input passage and the question regarding it.

**P:** **Viruses, bacteria, and fungi** can all cause pneumonia. In the United States, common causes of viral pneumonia are influenza and respiratory syncytial virus. A common cause of bacterial pneumonia is Streptococcus pneumonia. However, clinicians are not always able to find out which germ caused someone to get sick with pneumonia.

**Q:** Who experience difficulties finding causes for pneumonia?

The answer found by ELMO is **Viruses, bacteria, and fungi**, which, indeed, is not correct. The correct answer is **clinicians**. MRC fails miserably here associating **virus, bacteria, and fungi** with **Who**. Also, MRC failed to match the question with the sentence “However, clinicians are not always able to find out which germ caused someone to get sick with pneumonia.”. The introduced discourse-aware model answers the question as “Clinicians are not always able to find out.”. Let us consider the discourse structure for the passage and the question to understand the influence of discourse knowledge while dealing with this example.

The DT for this passage is shown in Fig. 3. In accordance with the constructed DT, we have a mapping between: Q: attribution → P: attribution, Q: cause → P: cause, Q: “causes” → P: “caused”. This information allows the model to attend each word to the relevant text spans in the input passage and, thus, to find the correct answer to the question.

![DT for Passage](image)

**Figure 3:** The discourse tree (DT) for text to chose an answer from (on the top) and for the question (on the bottom) with the mappings between corresponding nodes.

5 Conclusion

In this paper, we analyzed various linguistically enriched deep neural models and assessed the influence of semantic, syntax, and discourse on their performance on MRC tasks. While, modern systems are usually linguistic-free or rely on some independent linguistic characteristic, such as syntax or semantic individually, we claim that their combination could provide even higher accuracy gain. We also introduce the approach to incorporate discourse structure into the transformer-based model, which has been proven to be necessary for answering complex multi-sentence questions.

We have shown that the combination of three additional features encoded into a neural MRC is able to answer lengthy and complex questions better than the linguistic-free models, even the ones fine-tuned on the observed datasets. The introduced discourse-aware MRC model outperformed standalone syntax-guided (Zhang et al., 2020b) and semantic-enhanced models (Zhang et al., 2020a) for all the observed datasets. Although our MRC system did not achieve state-of-the-art results on some of the evaluation datasets (e.g., on the SQuAD), it demonstrated the superiority of integrated syntax/semantic/discourse subsystems in multiple diverse QA domains with complex questions.
References


Boris A. Galitsky, Sergei O. Kuznetsov, and Daniel Usikov. 2013. Parse thicket representation for multi-sentence search. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 7735 LNCS.


Amir Zeldes. 2016. rstWeb - a browser-based annotation interface for rhetorical structure theory and discourse relations.
A Dynamic Head Importance Computation Mechanism for Neural Machine Translation

Akshay Goindani Manish Shrivastava
International Institute of Information Technology - Hyderabad
{akshay.goindani@research.iiit.ac.in, m.shrivastava@iiit.ac.in}

Abstract
Multiple parallel attention mechanisms that use multiple attention heads facilitate greater performance of the Transformer model for various applications e.g., Neural Machine Translation (NMT), text classification. In multi-head attention mechanism, different heads attend to different parts of the input. However, the limitation is that multiple heads might attend to the same part of the input, resulting in multiple heads being redundant. Thus, the model resources are under-utilized. One approach to avoid this is to prune least important heads based on certain importance score. In this work, we focus on designing a Dynamic Head Importance Computation Mechanism (DHICM) to dynamically calculate the importance of a head with respect to the input. Our insight is to design an additional attention layer together with multi-head attention, and utilize the outputs of the multi-head attention along with the input, to compute the importance for each head. Additionally, we add an extra loss function to prevent the model from assigning same score to all heads, to identify more important heads and improvise performance. We analyzed performance of DHICM for NMT with different languages. Experiments on different datasets show that DHICM outperforms traditional Transformer-based approach by large margin, especially, when less training data is available.

1 Introduction
Transformer based NMT systems perform well on multiple translation tasks (Vaswani et al., 2017). Multi-head attention is a very important component of the Transformer model (Vaswani et al., 2017). Multiple heads improve performance compared to a single head, as they allow the model to jointly look at different subspaces, and hence capture enhanced features from sentences. For example, a head can capture positional information by attending to adjacent tokens, or it can capture syntactic information by attending to tokens in a particular syntactic dependency relation (Voita et al., 2019). However, the performance of the transformer-base model with 8 heads at each layer is only 1 BLEU point higher than that of a similar model with just a single head at each layer (Voita et al., 2019). This is due to the fact that majority of the heads learn similar weights, and therefore, multiple heads attend to the same parts of the input. Hence, most of the heads are redundant, leading to an increased computational complexity without improving performance.

To avoid this redundancy, one approach is to prune the redundant heads based on certain importance score. In this work, we focus on designing an importance computation method to compute the importance score for each head. Some recent work has analyzed the importance of heads by considering average attention weights of each head at some specific position (Voita et al., 2018). However, average of attention weights is a static measure of the head importance as it does not consider the varying importance of each head with respect to the input. The importance of a head is dynamic, as a head can be very important for a particular word, but can be less important for other words. Thus, in this work, we propose a Dynamic Head Importance Computation Mechanism (DHICM) to calculate the importance score for each head, and this can be later utilized to design a pruning strategy. Our key idea is to apply a second level attention on the outputs of all heads, to dynamically calculate the importance score for each head, that varies with the input, while training. We also propose to add a new loss term to prevent our approach from assigning equal importance to all heads.
Note that we apply DHICM for both self attention heads and encoder-decoder attention heads present in the encoder and decoder of the transformer architecture.

To evaluate the performance of our method, we considered multiple translation tasks with different language pairs such as Hindi-English, Belarusian-English, German-English. Results show that DHICM achieves a much higher performance compared to the standard transformer model, particularly, in low-resource conditions where much less training data is available. Moreover, DHICM requires only $\sim d^2$ additional parameters ($d$ is the word embedding dimension), that is much less than the total number of parameters in the transformer base model. The transformer model has a large number of hyperparameters, due to which, it is computationally challenging to search for their optimal values. Thus, much of the previous work used default values of the hyperparameters (Gu et al., 2018; Aharoni et al., 2019). However, these are not guaranteed to yield optimal performance on different datasets. Grid search over all hyperparameters is computationally intensive due to the exponential number of combinations across all possible values. Therefore, in this work, we perform grid search over a subset of hyperparameters, i.e., architecture hyperparameters and regularisation hyperparameters, and experiments show that the hyperparameter values obtained from our method yield significantly better performance compared to the default values. To summarize, our work makes the following major contributions:

- We propose a Dynamic Head Importance Computation Mechanism for transformer based NMT systems, to compute the importance scores for all heads dynamically with respect to an input token.

- We propose to add an additional loss function that helps to compute different attention for different heads, and filter the most important heads.

- Our hyperparameter tuning method yields significantly better performance than the default values.

2 Background

2.1 Single-Head Attention

Given a sequence of $N$ $d$-dimensional vectors $X = (x_1, x_2, ..., x_N)$ and a query vector $y \in \mathbb{R}^d$, a single-head attention is a weighted aggregate of $x_i, i \in \{1, 2, ..., N\}$, followed by a linear transformation. The weights are obtained using a function $F(x_i, q)$ e.g., multi-layer perceptron (Bahdanau et al., 2014) or scaled dot product (Vaswani et al., 2017), and the attention $A_h(X, y|W_v, W_o)$ is computed as $A(X, y) = W_o \sum_{i=1}^{N} F(x_i, y) W_v x_i$, where $W_o$ and $W_v$ are learnable weights. In a transformer based NMT system, there is an encoder and a decoder. The encoder encodes the input sequence of tokens and outputs a sequence of vectors $X$. The decoder uses $X$ to generate a sequence of tokens. If the query vector $y$ is generated using the encoder, then the computed attention is known as self-attention. Whereas if the query vector $y$ is generated from the decoder, then the computed attention is known as encoder-decoder attention.

2.2 Multi-Head Attention

Multi-head attention mechanism runs through multiple single head attention mechanisms in parallel (Vaswani et al., 2017). Let there be a total of $H$ heads, where each head $h \in \{1, 2, ..., H\}$ corresponds to an independent single head attention. The output of each head $A_h(X, y|W_v^h, W_o^h)$ is calculated independently, and the final output of multiple heads is calculated using the outputs of all heads, i.e., $\Sigma_{h=1}^{H} A_h(X, y|W_v^h, W_o^h)$, where, $W_v^h, W_o^h$ are learnable weights for each head $h$.

3 Approach

3.1 Dynamic Head Importance Computation Mechanism (DHICM)

In the traditional transformer model, the output of the multi-head attention is a linear transformation over the concatenation of outputs of all heads. Therefore, the outputs of all heads have equal contribution. However, since all heads are not equally important to the input (Sec. 1), we propose to compute the importance of each head with respect to the input dynamically.

Our idea is that an additional attention layer will allow the model to pay more attention to the head that is more important to the input.
Thus, we design a second level attention that uses the input and output of all heads to compute attention scores, i.e., importance for all the heads with respect to the input, described as follows. Let $x \in \mathbb{R}^d$ be a $d$-dimensional input to the multi-head attention module, and $O^h$ be the output of head $h \in \{1, 2, ..., H\}$ (without applying the linear transformation $W^h$ described in Sections 2.1 and 2.2). We first learn a function $G(x, O^h)$ to determine the attention, i.e., importance score for head $h$. To approximate $G(x, O^h)$, we considered both multi layer perceptron and scaled dot product. In our experiments, we observed that both achieve similar performance, and since scaled dot product requires less number of parameters, we used the latter to compute $G(x, O^h)$:

$$ G(x, O^h) = \frac{\exp(s(x, O^h))}{\sum_{h=1}^{H} \exp(s(x, O^m))} \quad (1) $$

where,

$$ s(x, O^h) = \frac{O^h_T W^T U x}{\sqrt{d_m}} \quad (2) $$

Here, $W \in \mathbb{R}^{d_m \times d_k}, U \in \mathbb{R}^{d_m \times d}$ are learnable parameters, and $d_k, d_m$ are scaling factors for the multi-head attention and second level attention, respectively. We also add a dropout layer (Srivastava et al., 2014) after computing $Ux$ in Equation 2. Next, we compute the output of the second-level attention layer (DHICM) using the attention scores for each head, as follows:

$$ DHICM(x, O) = W_s \sum_{h=1}^{H} G(x, O^h)V O^h \quad (3) $$

where, $O = (O^1, O^2, ..., O^H)$, and $V \in \mathbb{R}^{d_m \times d_m}$ are learnable parameters. The output of the second level attention is then passed to the feed forward network. Note that DHICM learns only $\sim d^2$ additional parameters corresponding to $W, U, W_s, V$ in the second layer added, and this is much less than the total number of parameters in the standard transformer model (typical value of $d$ is 512).

**Objective** Let $L_c$ represent the cross entropy loss that is minimized to ensure that the model generates accurate tokens. However, by only considering $L_c$ as the objective, it might be possible that the model learns equal values of $G(x, O^h)$ for all $h \in \{1, 2, ..., H\}$. This would indicate that all heads are equally important to the input $x$, and thus, prevent us from filtering the most important heads. To avoid this, we add an extra loss term to penalize the model if the value of $G(x, O^h)$ becomes equal for all $h \in \{1, 2, ..., H\}$. More formally, let $a \in R^H$ be a vector representing the importance score of all heads according to the model, where $a_h = G(x, O^h)$ is the importance score of head $h$. Let $b \in R^H$ be a vector representing equal importance of all heads, i.e., $b_h = \frac{1}{H}$, where $H$ is a constant. Both $a$ and $b$ denote the importance distribution of the heads, where $a$ is learned by the model using the second level attention, and $b$ is a uniform distribution with equal importance for all heads. In order to avoid the model from assigning equal importance to all the heads, we maximize the Kullback-Leibler divergence (KL Divergence) between distributions $a$ and $b$. Note that both the distributions sum up to 1, i.e., $\sum_{h=1}^{H} a_h = 1$, and $\sum_{h=1}^{H} b_h = 1$, and that $a_h > 0, b_h > 0$ for all $h \in \{1, 2, ..., H\}$. Specifically, we add an extra loss term $L_{KL}$ as the KL Divergence between $a$ and $b$, given as:

$$ L_{KL}(a||b) = \sum_{h=1}^{H} a_h \ln \frac{a_h}{b_h} \quad (4) $$

The overall loss $L$, where we minimize $L_c$ and maximize $L_{KL}$, is computed as:

$$ L = L_c - \lambda \cdot L_{KL} \quad (5) $$

where $\lambda$ is a hyperparameter used to control the effect of $L_{KL}$ on the overall loss $L$. The objective is to minimize the overall loss $L$.

### 4 Experiment

#### 4.1 Dataset Description

We used German-English (De-En) parallel corpus obtained from IWSLT14 (Cettolo et al., 2014) and

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>IWSLT14</td>
<td>160K</td>
<td>7.3K</td>
<td>6.7K</td>
</tr>
<tr>
<td>WMT17-CS</td>
<td>5.9M</td>
<td>3K</td>
<td>6K</td>
</tr>
<tr>
<td>HindEnCorp</td>
<td>256K</td>
<td>7K</td>
<td>7K</td>
</tr>
<tr>
<td>TED talks (Be-En)</td>
<td>4.5K</td>
<td>1K</td>
<td>2.6K</td>
</tr>
</tbody>
</table>

Table 1: Train, Validation and Test split size for different datasets used in our experiments
<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Default</th>
<th>Optimal De-En</th>
<th>Optimal Hi-En</th>
<th>Optimal Be-En</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feed forward dim.</td>
<td>2048</td>
<td>2048</td>
<td>1024</td>
<td>128</td>
</tr>
<tr>
<td>Attention heads</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.1</td>
<td>0.5</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td>Attention Dropout</td>
<td>0.0</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Activation Dropout</td>
<td>0.0</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Dropout (Section 3.1)</td>
<td>N/A</td>
<td>0.5</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Label Smoothing</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Table 2: Default and Optimal Hyperparameters

WMT17 (Bojar et al., 2017) shared translation tasks to evaluate the performance of our proposed method. Table 1 reports the number of parallel sentences in training, validation and test splits of different datasets that are considered in our experiments. To compare with (Iida et al., 2019), we used WMT17 De-En training corpus as training set and newstest13 as validation set. Similar to (Iida et al., 2019), we concatenated newstest14 and newstest17 to make one test set. We call this WMT17 dataset with the modified test set as WMT17-CS dataset. To assess the performance of our method for low resource language pairs, we used Hindi-English (Hi-En) parallel corpus obtained from HindEnCorp0.5 (Bojar et al., 2014). Also, we created smaller training sets from the complete IWSLT14 training set. We randomly sampled 10K, 20K, 30K, 40K, 80K, 120K and 160K sentence pairs from the full training data. The validation and test datasets were the same across all training sets. We also evaluated the performance of our method on extremely low resource language pairs. We used Belarusian-English (Be-En) parallel corpus from TED talks (Qi et al., 2018) that contains only 4.5K parallel sentences in the training set. The HindEnCorp0.5 dataset contains 270K sentence pairs, out of which we randomly sampled 7K sentence pairs each for validation and test sets, and used the remaining sentences as the training set. We used moses toolkit (Koehn et al., 2007) to tokenize German, Belarusian and English sentences, and IndicNLP Library\(^1\) to tokenize Hindi sentences. For open-vocabulary translation, we segmented words using byte-pair encoding (BPE)\(^2\) (Sennrich et al., 2015). For Be-En parallel corpus, we learned 5K merge operations for both Be and En separately. For other datasets, we combined the source and target sentences of the training set for learning BPE. We learned 10K merge operations for IWSLT14 dataset, and 20K merge operations for other datasets.

4.2 Hyperparameter Optimization

The transformer model has a large number of hyperparameters, and hence the total number of combinations of possible values for these hyperparameters is exponential. Therefore, although the language pairs are different from the original pairs used to determine the default values, much of the previous work uses the default hyperparameters (e.g., (Gu et al., 2018; Aharoni et al., 2019)). However, different languages have different characteristics, and using the hyperparameters tuned for one language pair, might not yield the optimal performance for another language pair. Furthermore, the amount of data available for training also affects the choice of hyperparameters. Hence, for each language pair, we perform extensive hyperparameter tuning to get better performance. Since there are exponential number of combinations, grid search is computationally very intensive, and random search is not guaranteed to yield optimal hyperparameters. Hence, we perform hyperparameter search using different values for a subset of hyperparameters. We majorly tune on two types of hyper-parameters - architecture hyper-parameters (e.g., number of attention heads, feed-forward dimension), and regularization hyper-parameters (e.g., dropout, attention dropout, activation dropout, label smoothing). The remaining hyper-parameters such as word embedding size, number of layers, for both

\(^1\)IndicNLP Library

\(^2\)https://github.com/rsennrich/subword-nmt
encoder and decoder are set to their default values (similar to (Vaswani et al., 2017)), and kept constant throughout the search. We first tune the architecture hyperparameters and keep the regularization hyperparameters constant with their default values. Next, we tune the regularization hyperparameters using the optimal values for architecture hyperparameters. Since we consider only a small subset of hyperparameters, the number of combinations are not exponential, and hence we are able to use grid search to tune the hyperparameters. The optimal hyperparameters chosen are the ones that correspond to the minimum loss on the validation set. Also, we use early stopping (described in Section 4.3) to prevent our model from overfitting. Although our hyperparameter tuning method does not guarantee a global optimum, we observe a substantial improvement over the default hyperparameters in our experiments (Section 5). The values of default and optimal hyperparameters obtained using our hyperparameter search, are reported in Table 2.

4.3 Experimental Setup and Baselines

We consider the Standard Transformer-base model (Vaswani et al., 2017) as a baseline, and for implementation, we used fairseq toolkit (Ott et al., 2019). We also analyzed the effect of applying our proposed approach DHICM to different layers of both encoder and decoder of the transformer model, and observed that applying the second level attention at the last layer of both encoder and decoder yields the best score.

We refer to the hyperparameters reported in the Standard Transformer-base model (Vaswani et al., 2017) as the Default Hyperparameters, and those obtained using our hyperparameter search described in Section 4.2 are referred to as the Optimal Hyperparameters. We trained all the models on 4 Nvidia GeForce RTX 2080 Ti GPUs. The number of layers of encoder and decoder was set to 6, number of tokens per batch was set to 8000, and the word embedding dimension \(d\) was set to 512. We used Adam optimizer (\(\epsilon = 10^{-6}, \beta_1 = 0.9, \beta_2 = 0.98\)) (Kingma and Ba, 2014) with a learning rate of \(5 \times 10^{-4}\). We used inverse square root learning rate scheduler with 4000 warmup steps, and used beam search with beam size of 5 for generating the sentences. In our proposed approach, we add two additional hyperparameters, that is, \(\lambda\) (described in Section 3.1), and a dropout in the second level attention (described in Section 3.1).

The optimal values for the dropout added are provided in Table 2, and we set \(\lambda\) as 0.1, for all experiments, corresponding to the minimum loss on the validation set. We save model checkpoints after every epoch and select the best checkpoint based on the lowest validation loss. In order to minimize overfitting, we stop training if the validation loss does not decrease for 10 consecutive epochs.

For training the models on smaller, randomly sampled training sets from the full IWSLT14 training set (Sec. 4.1), we used the optimal hyper-parameters learned using the full IWSLT14 training set. We used BLEU (Papineni et al., 2002) as the evaluation metric to compare the performance of our approach with two versions of the baseline model, (i) T-base, which is the Transformer-base model trained using Default hyperparameters, and (ii) T-optimal, which is the Transformer-base model trained using Optimal hyperparameters (Sec. 4.2). Please note that, for all our experiments, the hyperparameters for T-optimal and DHICM are same.

5 Results

Table 3 shows the performance of different methods. We observe that T-optimal outperforms T-base, and this demonstrates that the optimal hyperparameters found in our extensive hyperparameter search yield higher performance compared to the default hyperparameters in (Vaswani et al., 2017). Also, DHICM achieves a higher BLEU score, and outperforms T-optimal on HindEnCorp and WMT17-CS datasets by 3.45 and 1.12 BLEU points, respectively. We also performed experiment on the extremely low resource language pair Be-En, and observed that T-base achieved 4.09 BLEU score, and

<table>
<thead>
<tr>
<th>Dataset</th>
<th>T-base</th>
<th>T-optimal</th>
<th>DHICM</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMT17-CS</td>
<td>21.33</td>
<td>24.56</td>
<td>25.68</td>
</tr>
<tr>
<td>HindEnCorp</td>
<td>17.3</td>
<td>22.96</td>
<td>26.41</td>
</tr>
<tr>
<td>TED talks (Be-En)</td>
<td>4.09</td>
<td>5.49</td>
<td>6.29</td>
</tr>
</tbody>
</table>

Table 3: BLEU Score of different models on WMT17-CS, HindEnCorp, and Be-En parallel-corpora (trained using full training set). Note that Be-En is an extremely low resource language pair.
Table 4: BLEU score averaged over 3 randomly sampled training sets from full IWSLT14 training set

<table>
<thead>
<tr>
<th>Train Set Size</th>
<th>T-base</th>
<th>T-optimal</th>
<th>DHICM</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K</td>
<td>9.23</td>
<td>4.39</td>
<td>14.03</td>
</tr>
<tr>
<td>20K</td>
<td>13.44</td>
<td>7.62</td>
<td>22.00</td>
</tr>
<tr>
<td>30K</td>
<td>16.43</td>
<td>23.06</td>
<td>26.37</td>
</tr>
<tr>
<td>40K</td>
<td>19.31</td>
<td>27.79</td>
<td>28.32</td>
</tr>
<tr>
<td>80K</td>
<td>27.58</td>
<td>32.73</td>
<td>32.93</td>
</tr>
<tr>
<td>120K</td>
<td>30.93</td>
<td>34.60</td>
<td>34.7</td>
</tr>
<tr>
<td>160K</td>
<td>32.72</td>
<td>35.85</td>
<td>35.92</td>
</tr>
</tbody>
</table>

T-optimal achieved 5.49 BLEU score. Thus, T-optimal outperformed T-base by 1.4 BLEU points. Moreover, DHICM achieved 6.29 BLEU score, thus outperforming T-optimal by 0.8 BLEU points. We also compared the performance of our method with the multi-hop multi-head attention model (Iida et al., 2019) on WMT17-CS De-En dataset. We observed that DHICM outperforms (Iida et al., 2019) by 1.77 BLEU points.

Table 4 shows the BLEU score achieved by the models trained with smaller training sets that are randomly sampled from full IWSLT 2014 training set. We observe that the performance of all methods increases with an increase in the training set size, and DHICM achieves a much higher performance compared to T-base for all training set sizes. The performance of T-optimal and DHICM is similar for larger datasets, however, for low-resource datasets, our approach outperforms T-optimal by a large margin.

Since the hyperparameters for both T-optimal and DHICM are same, we can see that the gain in the performance of our method is due to the proposed second layer attention over the multi-head attention. In addition, our proposed loss function (Section 3.1) prevents the model from assigning the same importance to all heads. Thus, we are able to filter more important heads.

6 Analysis

Our proposed approach DHICM outperforms T-base and T-optimal by a large margin in the low resource conditions. We further analyzed the performance of the baseline model and DHICM, and observed that DHICM learns better word alignment especially, in low resource conditions. One of the reasons for learning better alignment can be that for each word, all heads are not equally important. The second level attention that we designed in our model allows the tokens to pay more attention to the heads that capture more relevant information for translation. Since the heads that are more relevant receive more attention, the parts of the input to which these heads attend, in turn receive more attention, and thus, the alignment becomes better. For example, providing more attention to the heads that capture the syntactic or semantic information, and relatively less attention to the heads that capture positional information. This justifies our hypothesis mentioned in Section 3.1.

We also verified this using the encoder-decoder attention distribution of the models shown in Figure 1 (low resource conditions) and Figure 2 (high resource conditions). The decoder of the transformer model uses the outputs of the encoder to generate the tokens in the target language. Each generated token pays some attention to each token in the source language. The attention distribution matrix shows the attention paid by the generated tokens in the target sentence (rows) to the tokens in the source sentence (columns). In Figure 1a and Figure 2a, we can see that most of the tokens on the source side get similar attention for the baseline approach. Moreover, the highest attention a source token receives is approximately 0.12 and 0.5 in Figure 1a and Figure 2a, respectively. This implies that the most important source token for translation does not receive enough attention, resulting in a poor word alignment. On the contrary, for DHICM (Figure 1b and Figure 2b), we observe a large variance in the distribution of the attention paid by a target token to the source tokens. Thus, more appropriate source tokens receive higher attention scores (∼0.8) in DHICM, leading to a better word alignment, as shown in both Figure 1b and Figure 2b. Also when 160K training sentences are used for IWSLT14, although the performance of the baseline and DHICM is similar, DHICM learns better word alignments compared to the baseline (shown in Figure 2), as DHICM helps the model to pay more attention to more relevant source tokens. Moreover, DHICM allows the model to pay higher attention (∼0.8) to the appropriate source words compared to the baseline model where highest attention received by a source token is ∼0.5. This shows that for both low resource and high resource conditions, DHICM helps the model to pay higher
attention to the more relevant source tokens.

We also analysed the additional attention layer introduced in DHICM. We compute the attention paid by each token to each head. Using the second level attention, we compute the attention paid by a particular token to all the heads and plot the attention values to create an attention distribution matrix. Figure 3 shows the attention distribution for the second level attention added on top of the multi-head self attention in the last layer of the encoder. The attention distribution matrix shows the attention paid by each source token (rows) to all the 4 heads (columns). The distribution shows that each token pays different amount of attention to each head, and this justifies our hypothesis that all heads are not equally important. Also, different tokens pay different amount of attention to a particular head, which also supports our hypothesis that the importance of a head is dynamic in nature, i.e., it varies as the input token changes. The attention distribution matrix also shows that the additional loss term indeed allows the model to compute different importance scores for different heads. In Figure 3, we can see that the second head gets the least attention from all the tokens. This shows that our proposed method identifies the least important heads, and thus, by incorporating DHICM, an appropriate pruning strategy can be developed to prune the least important heads.

7 Related Work

Some recent work has shown that most of the heads in a multi-head attention model become
redundant during test time (Michel et al., 2019). (Voita et al., 2018, 2019) analyzed the heads in a multi-head attention model, based on some importance score that is calculated after the model is fully trained. In contrast, in this work, we propose to calculate the importance scores dynamically while training.

A recent work (Iida et al., 2019) proposed to apply attention on top of the output of multi-head attention. However, they apply an additional attention layer only on the encoder, whereas, in our proposed method, we apply the second level attention on both encoder and decoder, that helps the generated target words to pay significant attention to appropriate source words, which in turn enhances the encoder-decoder attention distribution as shown in Figure 1b. Moreover, their proposed approach might learn equal attention weights for the additional attention layer, which would make all the heads equally important. In such a case, their approach would perform similar to transformer base model, even after adding more number of parameters compared to the standard transformer. To address this, we add an extra loss term in our method, to penalize for learning similar weights for the second level attention. This helps our method to compute different importance scores for different heads. Furthermore, during the calculation of the final attention, they transform the output of each head using a different transformation matrix for each head, while our proposed approach DHICM uses a single transformation matrix for the outputs of all heads. Thus, DHICM learns much fewer number of parameters in addition to achieving greater performance (the number of additional parameters learned in their approach is 550K, whereas DHICM learns 500K additional parameters).

8 Conclusion and Future Work

In this work, we proposed an effective Dynamic Head Importance Computation Mechanism (DHICM) to dynamically calculate the importance of different heads during training. Our idea is to calculate the importance with an additional attention layer along with the standard multi-head attention. We also proposed a loss function to prevent our method from computing equal importance for all heads, which together with the second-level attention facilitates to dynamically identify heads that are most important to the input word. Thus, the target words generated pay significantly high attention to the more appropriate/relevant source words. We also performed extensive hyperparameter tuning on a subset of hyperparameters, and observed that the optimal hyper-parameters obtained from our search yield a much higher BLEU score compared to the default hyper-parameters. Experiments on multiple translation tasks show that DHICM outperforms the standard transformer model by a large margin, especially in low resource settings. In the future, we will use the importance scores of the heads computed using DHICM and implement a strategy for pruning the less important heads. We would also like to explore further in the direction of reducing redundancy in multi-head attention.
References


Ondřej Bojar, Vojtěch Diatka, Pavel Stracha, Aleš Tamchyna, and Daniel Zeman. 2014. HindEnCorp 0.5. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (UFAL), Faculty of Mathematics and Physics, Charles University.


Henry Gorelick¹, Biddut Sarker Bijoy*², Syeda Jannatus Saba², Sudipta Kar³, Md Saiful Islam²,⁴, Mohammad Ruhul Amin¹

¹Fordham University
²Shahjalal University of Science and Technology
³Amazon Alexa AI
⁴University of Alberta

hgorelick@fordham.edu, {biddut12, syeda06}@student.sust.edu, skar3@uh.edu, mdsaiful@ualberta.ca, mamin17@fordham.edu

Abstract

In this paper, we attempt to improve upon the state-of-the-art in predicting a novel’s success by modeling the lexical semantic relationships of its contents. We created the largest dataset used in such a project containing lexical data from 17,962 books from Project Gutenberg. We utilized domain specific feature reduction techniques to implement the most accurate models to date for predicting book success, with our best model achieving an average accuracy of 94.0%. By analyzing the model parameters, we extracted the successful semantic relationships from books of 12 different genres. We finally mapped those semantic relations to a set of themes, as defined in Roget’s Thesaurus and discovered the themes that successful books of a given genre prioritize. At the end of the paper, we further showed that our model demonstrate similar performance for book success prediction even when Goodreads rating was used instead of download count to measure success.

1 Introduction

Since its publication in 1868, approximately 1.78 million copies of Louisa May Alcott’s Little Women have been sold, which equates to about 1,000 copies a month for 152 years. Every publisher in the industry hopes to find a manuscript that can sell even 10,000 copies in its lifetime. This begs the question: what makes Little Women a timeless success? Recently, researchers have attempted to use machine learning and natural language processing to answer this question, among others.

Predicting the success of a novel by analyzing its content is a challenging research problem. Thousands of new books are published every year, and only a fraction of them achieve wide popularity. Therefore, the ability to predict a book’s success prior to publication would be exceptionally useful to the publishing industry and enable editors to make better decisions. Many factors contribute to a book’s success including, but not limited to plot, setting, character development, etc. Additionally, there are some other factors that contribute to a book’s popularity that an author and publisher cannot control like the time when a book is published, the author’s reputation, and the marketing strategy. In this paper, we only focus on the content of the book to predict its popularity.

In this paper, we explore whether novels in a spe-
cific genre have certain dominant themes in common based on stylometric features, and if so, what meaning we can attribute to those themes as it relates to a book’s success. To attain this objective, we investigated ways to enable using the entire book’s content for stylistic modeling using frequencies of lexical production rules of CFGs for each novel (see Figure 1) followed by semantic word association of those rules to Roget’s Categories and Themes for better interpretation. In this work, we followed widely used feature reduction technique for SVM modeling to reduce the large lexical feature space for lengthy novels and experiment with different techniques using POS, Unigram, WordNet, and lexical production rules. In this article we present the following contributions:

- We built the largest dataset containing a total of 17,962 books. We included books from 4 additional genres and reclassified 2 of the genres included in (Ashok et al., 2013) as follows: Mystery→Detective; Love→Romance.

- We introduced our feature reduction methods to greatly improve prediction performance with our best model achieving 94% accuracy for success prediction.

- We mapped both WordNet’s semantic word relations and context free grammar rules to a set of themes, as defined in Roget’s Thesaurus. With these mappings, we discovered the themes that successful books of a given genre prioritize.

2 Related Works

Roget’s Thesaurus is a widely used English-language thesaurus. A British lexicographer, Peter Mark Roget (1779–1869), created the thesaurus in 1805. The first version of the thesaurus comprised of nearly 15,000 words and was released to the public on 29 April 1852 (Roget and Roget, 1886). Since then each successive edition was improved with more words, with the most recent edition containing more than 100,000 words. In previous work, Jarmasz and Szpakowicz (2004) showed that Roget’s is an excellent resource for measuring semantic similarity and Roget’s word clusters have higher correlation than many other prominent word groups e.g., Wordnet (Miller, 1998; Jarmasz, 2012).

Syntactic features, such as CFG productions have been found to be very effective in different NLP tasks. Raghavan et al. (2010) used CFGs for authorship attribution achieving very high accuracy such as 96%. Rayson et al. (2002) presented systematic analyses based on lexical and syntactic features for genre detection of a literary works showing that novels involve more use of verbs and adverbs. On the other hand, Douglas and Brussard (2000) showed that informative writing tend to use nouns, prepositions, determiners and coordinating conjunctions more. CFGs were also used in several other works, such as gender attribution by tracing stylometric evidence by (Sarawgi et al., 2011), and native language detection by exploiting parse structures (Wang and Zong, 2011).

In the earlier work, Ashok et al. (2013) used stylistic approaches, such as unigram, bigram, part-of-speech distribution, grammatical rules, constituents, sentiment, and connotation as features and used Liblinear SVM (Fan et al., 2008) for the book success classification task. They used books from 8 genres, and they were able to achieve an average accuracy of 73.5% across all genres. Maharjan et al. (2017) used a set of hand-crafted features in combination with a recurrent neural network and generated feature representation to predict success. They obtained an average F1-score of 73.5% for 8 genres. In a more recent work by Maharjan et al. (2018a), they used the flow of emotion throughout a book for success prediction and obtained an F1-score of 69%.

In this paper, we used widely used feature reduction technique for SVM modeling. Guyon et al. (2002) used SVM weights for assigning ranks in the feature selection process. They verified that the top-ranked genes found by SVM have biological relevance to cancer and the SVM classifier with SVM selected features worked better than other classifiers in determining the relevant features along with the classification task.

3 Dataset Construction

3.1 Original Dataset

The original dataset from Ashok et al. (2013) is quite small as it only includes the first 1,000 sentences from 800 books split into 8 different genres, which are further split into successful and unsuccessful classes, each having 50 books. Additionally, many of the files included have less than 1,000 sentences, or contain automatically generated text from Project Gutenberg instead of the text from the proper novel. Finally, the books included are prelabeled with their successful/unsuccessful class.
where download counts are absent, which limits further testing. Considering these issues, we decided to build upon (Ashok et al., 2013) by creating a cleaner and more complete dataset. Additionally, we present multiple models that are both more accurate and more general than the best performing model in (Ashok et al., 2013), unigram. From these models, we discovered more interesting and revealing qualities that separate successful from non-successful books.

3.2 New Dataset

We downloaded and used 17,962 English novels from Project Gutenberg: an online catalog of over 60,000 books, which are available to download for free in various formats (Gutenberg). We filtered the 60k books as follows: a) only English books, and b) only fiction books. We used a bash script\(^1\) to harvest the novels from Project Gutenberg according to the webmaster’s guidelines\(^2\).

After downloading the books, we used the NLTK API for data processing (Bird et al., 2009). For each book, we extracted the unigram and bigram frequencies, the part-of-speech (POS) tag using the Stanford CoreNLPParser frequencies, the lexical and non-lexical context free grammar production rules also using the Stanford CoreNLPParser, the Roget’s Thesaurus Category frequencies, and the WordNet Synset frequencies (Roget, 1852; Prince-

We utilized 12 linguistic models for our quantitative analysis. 6 of the models are our own implementation of models used in (Ashok et al., 2013). Our 6 additional models have not been used to make these types of qualitative conclusions until now. These models include WordNet (Princeton University, 2010), Roget’s Thesaurus (Roget, 1852), two models that map WordNet to different levels of Roget’s Thesaurus, and two models that map context free grammar rules to Roget’s Thesaurus. Mapping examples are given in Table 2 and explained below.

\begin{table}[h]
\centering
\begin{tabular}{|l|c|c|c|}
\hline
\textbf{GENRE} & \textbf{# BOOKS} & \textbf{\(v^<\)} & \textbf{\(v^>\)} \\
\hline
Adventure & 917 & 28 & 46 \\
Children & 3278 & 27 & 35 \\
Detective & 285 & 41 & 74 \\
Drama & 785 & 45 & 62 \\
Fantasy & 382 & 76 & 81 \\
Fiction & 5369 & 22 & 38 \\
Historical Fiction & 961 & 32 & 50 \\
Humor & 1024 & 14 & 24 \\
Poetry & 1664 & 34 & 50 \\
Romance Fiction & 634 & 34 & 48 \\
Science Fiction & 1748 & 44 & 58 \\
Short Stories & 915 & 35 & 49 \\
All & 17,962 & 35 & 37 \\
\hline
\end{tabular}
\caption{\# of novels per genre and download count thresholds for unsuccessful (\(\leq v^<\)) and successful (\(\geq v^>\)) classes.}
\end{table}

4 Methodology

4.1 Linguistic Models

We utilized 12 linguistic models for our quantitative analysis. 6 of the models are our own implementation of models used in (Ashok et al., 2013). Our 6 additional models have not been used to make these types of qualitative conclusions until now. These models include WordNet (Princeton University, 2010), Roget’s Thesaurus (Roget, 1852), two models that map WordNet to different levels of Roget’s Thesaurus, and two models that map context free grammar rules to Roget’s Thesaurus. Mapping examples are given in Table 2 and explained below.

**Unigram:** The frequency of unique words in text.

**Part-of-Speech Distribution:** The authors of Ashok et al. (2013) demonstrated the value of PoS tag distribution in success prediction, and Koppel et al. (2006) presented the relationship between PoS tagging and genre detection and authorship attribution. Therefore, we reevaluated the application of PoS tag distribution for success prediction.

**Context Free Grammar Rule Distribution:** We also reevaluate the analysis of CFG rule distribution as presented in (Ashok et al., 2013), and use the same four categories:

- \(\Gamma\): lexical production rules (productions where the right-hand symbol (RHS) is a terminal symbol (word)).
Table 2: Mapping to Roget examples for WordNet and $Γ^G$. For each model, the **ORIGINAL FEATURES** are combined in the **ROGET CATEGORY** column, which in turn is combined in the **ROGET THEME** column.

<table>
<thead>
<tr>
<th>Model</th>
<th>Original Feature</th>
<th>Roget Category</th>
<th>Roget Theme</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordNet</td>
<td>blaze, glitter, sunny light</td>
<td>Nom$\rightarrow$Adj$\rightarrow$life</td>
<td>Organic Matter</td>
</tr>
<tr>
<td>$Γ^G$</td>
<td>Nom$\rightarrow$Adj$\rightarrow$bad Nom$\rightarrow$Adj$\rightarrow$illegal Nom$\rightarrow$Adj$\rightarrow$lawful Nom$\rightarrow$Adj$\rightarrow$unconstitutional</td>
<td>Nom$\rightarrow$Adj$\rightarrow$wrong Nom$\rightarrow$Adj$\rightarrow$wrong Nom$\rightarrow$Adj$\rightarrow$wrong Nom$\rightarrow$Adj$\rightarrow$wrong</td>
<td>Nom$\rightarrow$Adj$\rightarrow$Moral</td>
</tr>
</tbody>
</table>

- **$Γ^G$**: lexical production rules prepended with the grandparent node.
- **$γ$**: nonlexical production rules (productions where the RHS is a non-terminal symbol).
- **$γ^G$**: nonlexical production rules prepended with the grandparent node.

**WordNet**: WordNet is large lexical database of English words. The WordNet database groups nouns, verbs, adjectives, and adverbs into sets of cognitive synonyms called Synsets. Each Synset expresses a distinct concept and is represented by a single word. Since Synsets represent conceptual synonyms, they are able to be linked through conceptual and semantic relationships (Princeton University, 2010). WordNet has a total of 117,659 Synsets, each represented by a single, unique word, and our model uses the frequencies of these Synsets in each book. Not only does WordNet fit our semantic relation analysis methodology, but it has been used for the relevant task of metaphor identification in (Mao et al., 2018).

**Roget’s Thesaurus**: A tree structured thesaurus with six root nodes, which we will refer to as Roget Classes or Classes for short. Each Class is divided in sections, which results in 23 total sections. These sections represent 23 unique concepts that are both general enough to encompass a wide range of ideas, but also specific enough to retain clear meaning. Therefore, we refer to these sections as Themes, and they are the critical piece to interpreting the results of class prediction. Themes are further divided into subsections, levels, etc. before terminating in 1,039 groups of synonyms, which we will refer to as Categories. The Categories are comprised of 56,769 total words, with about half appearing in multiple Categories (Roget, 1852). Our Roget model uses the frequencies of these Categories in each book. Furthermore, the authors of (Aman and Szpakowicz, 2008) demonstrated the possible applications of Roget’s Thesaurus for emotion detection with natural language processing, and (Kennedy and Szpakowicz, 2010) used the thesaurus for the related process of text summarizing.

**Mapping WordNet to Roget**: Since Roget’s Thesaurus has fewer synonym groups than WordNet (1,039 vs. 117,659), and those groups are hierarchically abstracted with each of the 1,039 Roget Categories belonging to one of the 23 Roget Themes, we mapped WordNet’s Synsets to Roget’s Thesaurus to discover more meaningful insights into the distinct characteristics of successful novels. We mapped WordNet to Roget Categories (WNRC), and then subsequently to Roget Themes (WNRT).

**Mapping Lexical Production Rules to Roget**: Since the RHS of lexical production rules are words, they can also be mapped to Roget’s Thesaurus. Using the RHS of the lexical production rules for each book we derived $Γ^G$ to Roget Categories ($Γ^G$RC) and subsequently to Roget Themes ($Γ^G$RT).

### 4.2 Implementation

We used the sci-kit learn implementation of LibLinear SVM with 5-fold cross validation for class prediction (Pedregosa et al., 2011; Fan et al., 2008). To tune the weighted linear SVM parameter C, we used the tool gridsearchCV (Pedregosa et al., 2011) and performed a search over the values ranging $1e(-4to3)$. Part-of-speech tag features are scaled with unit normalization, while all other features are scaled using tf-idf. We used two strategies for the class prediction task: predicting class by genre and predicting class independent of genre. We chose this model over neural models as it gives us better scope to interpret book success with hand crafted features. After the initial training and testing of each model, we employed an exhaustive feature reduction method, similar to our success labeling process, to maximize performance (see Figure 2).

For a given model, we start with the mean feature weight learned during training. We remove all features from the dataset with $|weight|$ less than the $|mean|$ feature weight. Next, we train and test the model on this reduced fea-
Figure 2: Feature reduction process: WordNet success prediction accuracy vs. number of features.

The prediction accuracy for each model across all books, and each model by genre, both before and after feature reduction are shown in Table 4 and Table 5, respectively. As illustrated in both settings, the performance of nearly every model improved after we reduced the features with $\gamma^G$ showing the largest improvement of an average of 24.3% when reduced by genre and WordNet improving the most by 16.1% when reduced independent of genre.

The best performing models are indicated in bold.

<table>
<thead>
<tr>
<th>MODEL</th>
<th>ACCURACY</th>
<th>ACCURACY$^R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unigram</td>
<td>61.6</td>
<td>61.6</td>
</tr>
<tr>
<td>POS</td>
<td>61.1</td>
<td>61.1</td>
</tr>
<tr>
<td>$\Gamma$</td>
<td>64.3</td>
<td>73.8</td>
</tr>
<tr>
<td>$\Gamma^G$</td>
<td>64.2</td>
<td>80.1</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>61.1</td>
<td>68.9</td>
</tr>
<tr>
<td>$\gamma^G$</td>
<td>59.5</td>
<td>71.5</td>
</tr>
<tr>
<td>Roget</td>
<td>65.3</td>
<td>66.2</td>
</tr>
<tr>
<td>WordNet</td>
<td>63.6</td>
<td>79.7</td>
</tr>
<tr>
<td>WNRC</td>
<td>67.6</td>
<td>68.8</td>
</tr>
<tr>
<td>WNRT</td>
<td>61.5</td>
<td>61.5</td>
</tr>
<tr>
<td>$\Gamma^{GRC}$</td>
<td>66.9</td>
<td>67.8</td>
</tr>
<tr>
<td>$\Gamma^{GRT}$</td>
<td>60.8</td>
<td>60.8</td>
</tr>
</tbody>
</table>

Table 4: Accuracy of prediction model for ALL BOOKS of new dataset, with original and optimal reduced feature set ($R$).

models were overfitting the dataset. This observation further motivated us to build a much larger dataset. For the classification task on newly constructed dataset, we applied the 5-fold cross validation method on all the genre specific datasets for evaluating each machine learning model. While for the feature reduction task, features were reduced using training weights from the training set, then tested on the test set. We had continued reducing the training set until the resulting features did not improve performance on the remaining test set.

5 Experimental Results

Using the original small dataset (Ashok et al., 2013), the prediction accuracy for each model by genre is presented in Table 8 at Appendix A1, and highlights another primary reason for increasing the size of the dataset. As each of the models was found to be achieving 100% accuracy in success prediction, we were convinced that those
ture set and record the accuracy. For each subsequent test, starting at a step value of 0.25, we take only the features with weights greater than or equal to $Mean(OriginalWeights) + (StdDev(OriginalWeights) * Step)$. This process continues, increasing the step value by 0.25 after each iteration, until one of the following conditions is met: 100% classification accuracy is achieved, maximum accuracy is found (determined if multiple consecutive subsequent feature sets produce decreasing performance), or the number of features is reduced to less than 1% of the original number of features. Additionally, as explained previously, the processes of mapping WordNet to Roget’s Thesaurus is a feature reduction technique in its own right. Table 3 illustrates the degree of feature reduction when WordNet and $\Gamma^G$ are mapped.

<table>
<thead>
<tr>
<th>MODEL</th>
<th># OF FEATURES</th>
<th># OF FEATURES$^R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordNet</td>
<td>31,833</td>
<td>1,670</td>
</tr>
<tr>
<td>WNRC</td>
<td>840</td>
<td>272</td>
</tr>
<tr>
<td>WNRT</td>
<td>21</td>
<td>9</td>
</tr>
<tr>
<td>$\Gamma^G$</td>
<td>24,302</td>
<td>596</td>
</tr>
<tr>
<td>$\Gamma^{GRC}$</td>
<td>995</td>
<td>184</td>
</tr>
<tr>
<td>$\Gamma^{GRT}$</td>
<td>21</td>
<td>13</td>
</tr>
</tbody>
</table>

Table 3: Number of features of ADVENTURE books before/after reduction for WordNet and $\Gamma^G$ models.
6 Interpreting Book Success Prediction

While our reduced $\Gamma^G$ and WordNet models display excellent performance in both test settings (by genre and independent of genre), the resulting feature sets are not self-explanatory. In other words, the respective lexical production rules and Synsets that the models deem most important do not necessarily highlight some interesting aspect of successful books. This is where Roget’s Thesaurus proves most valuable.

We figured that if we looked up the Roget Theme of the RHS for each lexical production rule and the Roget Theme for each WordNet Synset we would find that the successful and unsuccessful books prioritize different Themes. With this hypothesis in mind, we mapped the reduced WordNet and reduced $\Gamma^G$ models to new $\Gamma^{GR}$ models by first looking up the Roget Category of each Synset and RHS, respectively, from the reduced feature sets, and then summing the frequencies in each group of Synsets/symbols. As we did with each previous model, we reduced the new WNRC and $\Gamma^{GR}$ models we mapped again, this time from $\Gamma^{GR}$ produced the highest accuracy for each genre except DETECTIVE. For DETECTIVE novels, $\gamma^{GR}$ outperforms all models with 100% accuracy.

Figure 2 illustrates the pattern of performance improvement that each model exhibits through the feature reduction process both by genre and independent of genre. As the number of features is reduced, the average accuracy for success prediction increases until the algorithm finds the best set of features and achieves peak performance. Then accuracy sharply drops as the feature set is reduced further. The fact that each model demonstrates such behavior validates the effectiveness of our feature reduction method.

### Table 5: Accuracy (% of classification results by Genre) for new dataset, with/without feature reduction (R).

<table>
<thead>
<tr>
<th>Model</th>
<th>Ad</th>
<th>Ch</th>
<th>Dr</th>
<th>Fa</th>
<th>Fi</th>
<th>Hi</th>
<th>Hu</th>
<th>Po</th>
<th>Ro</th>
<th>Sc</th>
<th>Sh</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uni</td>
<td>72.3</td>
<td>63.8</td>
<td>80.6</td>
<td>60.8</td>
<td>67.5</td>
<td>62.5</td>
<td>63.5</td>
<td>73.2</td>
<td>64.8</td>
<td>62.9</td>
<td>63.2</td>
<td>60.1</td>
</tr>
<tr>
<td>UniR</td>
<td>76.9</td>
<td>71.6</td>
<td>84.7</td>
<td>69.4</td>
<td>73.3</td>
<td>62.5</td>
<td>69.1</td>
<td>81.9</td>
<td>73.3</td>
<td>67.1</td>
<td>78.1</td>
<td>67.2</td>
</tr>
<tr>
<td>POS</td>
<td>66.2</td>
<td>63.8</td>
<td>72.7</td>
<td>63.1</td>
<td>67.0</td>
<td>66.0</td>
<td>70.5</td>
<td>71.6</td>
<td>70.3</td>
<td>60.1</td>
<td>63.2</td>
<td>68.5</td>
</tr>
<tr>
<td>POSR</td>
<td>66.2</td>
<td>63.9</td>
<td>73.5</td>
<td>63.8</td>
<td>69.4</td>
<td>66.2</td>
<td>70.7</td>
<td>77.6</td>
<td>70.6</td>
<td>63.6</td>
<td>63.8</td>
<td>69.4</td>
</tr>
<tr>
<td>$\Gamma$</td>
<td>74.6</td>
<td>66.5</td>
<td>85.5</td>
<td>66.0</td>
<td>68.5</td>
<td>68.9</td>
<td>72.4</td>
<td>78.7</td>
<td>70.0</td>
<td>65.4</td>
<td>62.0</td>
<td>69.5</td>
</tr>
<tr>
<td>$\Gamma^R$</td>
<td>95.0</td>
<td>86.5</td>
<td>99.6</td>
<td>89.7</td>
<td>92.5</td>
<td>87.8</td>
<td>92.8</td>
<td>97.4</td>
<td>88.1</td>
<td>95.7</td>
<td>90.7</td>
<td>91.6</td>
</tr>
<tr>
<td>$\Gamma^{GR}$</td>
<td>74.9</td>
<td>67.0</td>
<td>85.9</td>
<td>67.0</td>
<td>69.3</td>
<td>67.8</td>
<td>71.2</td>
<td>80.6</td>
<td>70.5</td>
<td>66.7</td>
<td>62.5</td>
<td>69.1</td>
</tr>
<tr>
<td>$\Gamma^{GR}$</td>
<td>96.9</td>
<td>88.5</td>
<td>99.6</td>
<td>94.1</td>
<td>96.2</td>
<td>88.3</td>
<td>93.5</td>
<td>95.8</td>
<td>89.8</td>
<td>97.0</td>
<td>92.8</td>
<td>95.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Theme</th>
<th>Ad</th>
<th>Ch</th>
<th>Dr</th>
<th>Fa</th>
<th>Fi</th>
<th>Hi</th>
<th>Hu</th>
<th>Po</th>
<th>Ro</th>
<th>Sc</th>
<th>Sh</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma$</td>
<td>69.9</td>
<td>59.7</td>
<td>78.9</td>
<td>62.3</td>
<td>65.9</td>
<td>60.9</td>
<td>66.1</td>
<td>73.5</td>
<td>65.7</td>
<td>62.1</td>
<td>56.5</td>
<td>66.8</td>
</tr>
<tr>
<td>$\gamma^R$</td>
<td>93.6</td>
<td>79.1</td>
<td>99.2</td>
<td>87.3</td>
<td>93.3</td>
<td>80.0</td>
<td>86.8</td>
<td>94.4</td>
<td>83.8</td>
<td>91.4</td>
<td>83.4</td>
<td>87.2</td>
</tr>
<tr>
<td>$\gamma^{GR}$</td>
<td>68.5</td>
<td>59.8</td>
<td>82.3</td>
<td>61.5</td>
<td>65.6</td>
<td>63.8</td>
<td>67.4</td>
<td>74.9</td>
<td>66.3</td>
<td>65.0</td>
<td>58.4</td>
<td>68.4</td>
</tr>
<tr>
<td>$\gamma^{GR}$</td>
<td>95.1</td>
<td>83.7</td>
<td>100.0</td>
<td>89.2</td>
<td>92.8</td>
<td>82.5</td>
<td>87.9</td>
<td>96.6</td>
<td>87.0</td>
<td>96.7</td>
<td>91.8</td>
<td>91.1</td>
</tr>
<tr>
<td>$\Gamma$</td>
<td>68.9</td>
<td>66.9</td>
<td>79.1</td>
<td>66.0</td>
<td>68.3</td>
<td>69.4</td>
<td>72.9</td>
<td>80.2</td>
<td>70.5</td>
<td>65.3</td>
<td>64.4</td>
<td>70.6</td>
</tr>
<tr>
<td>$\Gamma^R$</td>
<td>81.5</td>
<td>71.2</td>
<td>91.1</td>
<td>75.8</td>
<td>82.1</td>
<td>72.7</td>
<td>78.2</td>
<td>84.0</td>
<td>74.6</td>
<td>77.3</td>
<td>70.5</td>
<td>79.0</td>
</tr>
<tr>
<td>WN</td>
<td>75.5</td>
<td>67.6</td>
<td>84.7</td>
<td>65.2</td>
<td>69.8</td>
<td>68.1</td>
<td>72.5</td>
<td>80.6</td>
<td>70.4</td>
<td>65.8</td>
<td>62.8</td>
<td>69.0</td>
</tr>
<tr>
<td>WN^R</td>
<td>94.1</td>
<td>86.1</td>
<td>99.6</td>
<td>90.3</td>
<td>93.1</td>
<td>87.5</td>
<td>92.3</td>
<td>95.3</td>
<td>84.0</td>
<td>94.7</td>
<td>88.6</td>
<td>89.5</td>
</tr>
<tr>
<td>WNRC</td>
<td>79.7</td>
<td>72.2</td>
<td>93.4</td>
<td>74.8</td>
<td>79.3</td>
<td>72.0</td>
<td>81.4</td>
<td>86.6</td>
<td>72.8</td>
<td>82.9</td>
<td>70.0</td>
<td>77.4</td>
</tr>
<tr>
<td>WNRC^R</td>
<td>90.2</td>
<td>76.1</td>
<td>97.5</td>
<td>86.3</td>
<td>93.1</td>
<td>75.2</td>
<td>90.3</td>
<td>93.2</td>
<td>78.4</td>
<td>92.4</td>
<td>76.3</td>
<td>85.0</td>
</tr>
<tr>
<td>WNRT</td>
<td>66.5</td>
<td>61.9</td>
<td>80.6</td>
<td>64.2</td>
<td>67.4</td>
<td>65.8</td>
<td>70.3</td>
<td>76.7</td>
<td>70.2</td>
<td>66.2</td>
<td>62.6</td>
<td>71.0</td>
</tr>
<tr>
<td>WNRT^R</td>
<td>68.0</td>
<td>62.7</td>
<td>82.7</td>
<td>64.9</td>
<td>68.4</td>
<td>65.9</td>
<td>71.2</td>
<td>77.4</td>
<td>70.5</td>
<td>68.7</td>
<td>63.3</td>
<td>72.3</td>
</tr>
<tr>
<td>$\Gamma$</td>
<td>86.1</td>
<td>84.0</td>
<td>92.4</td>
<td>81.2</td>
<td>87.4</td>
<td>74.5</td>
<td>82.8</td>
<td>88.0</td>
<td>76.2</td>
<td>87.3</td>
<td>74.4</td>
<td>78.9</td>
</tr>
<tr>
<td>$\Gamma^R$</td>
<td>92.9</td>
<td>77.6</td>
<td>98.8</td>
<td>89.4</td>
<td>95.4</td>
<td>77.9</td>
<td>90.7</td>
<td>93.6</td>
<td>82.2</td>
<td>95.4</td>
<td>79.8</td>
<td>87.3</td>
</tr>
<tr>
<td>$\Gamma^{GR}$</td>
<td>75.5</td>
<td>63.2</td>
<td>76.7</td>
<td>63.6</td>
<td>62.3</td>
<td>66.2</td>
<td>73.5</td>
<td>79.3</td>
<td>70.3</td>
<td>6701</td>
<td>62.3</td>
<td>71.0</td>
</tr>
<tr>
<td>$\Gamma^{GR}$</td>
<td>76.0</td>
<td>63.5</td>
<td>77.7</td>
<td>64.9</td>
<td>65.3</td>
<td>66.4</td>
<td>74.0</td>
<td>79.3</td>
<td>70.5</td>
<td>68.1</td>
<td>62.8</td>
<td>71.0</td>
</tr>
</tbody>
</table>

In Table 4 and Table 5. When predicting novel success by genre and independent of genre, $\Gamma^{GR}$ shows the best results predicting a book’s success class with an accuracy of 94.0% and 80.1%, respectively. Furthermore, when predicting success by genre, $\Gamma^{GR}$ achieves the highest accuracy for each genre except DETECTIVE. For DETECTIVE novels, $\gamma^{GR}$ outperforms all models with 100% accuracy.

We figured that if we looked up the Roget Theme of the RHS for each lexical production rule and the Roget Theme for each WordNet Synset we would find that the successful and unsuccessful books prioritize different Themes. With this hypothesis in mind, we mapped the reduced WordNet and reduced $\Gamma^G$ models to new $\Gamma^{GR}$ models by first looking up the Roget Category of each Synset and RHS, respectively, from the reduced feature sets, and then summing the frequencies in each group of Synsets/symbols. As we did with each previous model, we reduced the new WNRC and $\Gamma^{GR}$ models. From the WNRC and $\Gamma^{GR}$ models we mapped again, this time from Roget Categories to the 23 Roget Themes, which produced the WNRT and $\Gamma^{GR}$ models. Mapping examples are given in Table 2 and its outcome is detailed in the Figure 3.

We did not expect the performance of the WNRC and $\Gamma^{GR}$ models, since they were conceived strictly as intermediary maps between WordNet/$\Gamma^G$ and Roget Themes. $\Gamma^{GR}$ produced the highest baseline results of all the models without any feature reduction used in our experiments with 81.9% average accuracy by genre. Furthermore, $\Gamma^{GR}$ accurately predicts success classification per genre.
at an average rate of 88.4%. What’s impressive about the accuracy of \(\Gamma^{GRCR}\), when compared to that of \(\Gamma^G\), is the large difference in number of features used in each model as shown when predicting DETECTIVE novels in Table 3.

With these impressive results from \(\Gamma^{GRCR}\), we expected \(\Gamma^{GRRT}\) and \(\Gamma^{GRT}\) to follow suit despite learning with a feature set of at most 13 features. However, this was not the case as \(\Gamma^{GRRT}\) predicts the success of a book by its genre with an average accuracy of only 69.9%. As previously stated, the motivation for the construction of WNRT and \(\Gamma^{GRRT}\) was strictly to find a common thread between successful novels in each genre. Therefore, the poor performance of the WNRT and \(\Gamma^{GRRT}\) models does not undercut the reasoning behind its conception, and the high accuracy of WordNet, WNRCR, \(\Gamma^G\), and \(\Gamma^{GRCR}\) supports our claim that each is a general model that can reveal underlying characteristics of successful books.

Additionally, WNRT and \(\Gamma^{GRRT}\) do not improve performance after feature reduction when classifying independent of genre. This outcome also supports our original hypothesis as it shows that the models require each of the 23 Roget Themes in order to make the most accurate prediction. The lack of improvement in WNTR and \(\Gamma^{GRRT}\) when predicting success class independent of genre also demonstrates the relationship between a novel’s genre and its prioritization of certain Themes.

### 6.1 Successful Categories and Themes for a Genre

Figure 4 illustrates the top 30 discriminative positive and negative Roget Categories based on the model weights for CHILDREN’s book success prediction. Greener Categories are positively weighted for success while redder Categories are negatively weighted. Specifically, we see positive weights for Themes of “Formation of Ideas” and “Related to Space.” These Themes align with what readers should expect from CHILDREN’s stories: developing new ideas (Formation of Ideas) as a character grows and has new experiences in the physical world around them (Related to Space). It’s interesting however that “Communication of Ideas” shows negative weight for this genre. This suggests that CHILDREN’s stories are more concerned with how a person grows and develops their own ideas, rather than how they communicate them. This pattern of the prioritization of expected Themes holds true across all genres, with few exceptions. Therefore, we can conclude that lexical choices focusing on Themes that conform to genre norms produce more successful novels.

Figure 3: This heatmap presents how the mapping of \(\Gamma^G\) to \(RT\) helps to interpret success of ADVENTURE books. The plot presents both +ve/-ve Roget Themes on the row, and successful/unsuccessful books on the column. Each cell represents the relative frequency of observing \(\Gamma^G\) in an \(RT\). We observe that authors of successful books used certain CFGs that result in higher frequency in +ve RT cells, while the unsuccessful books have higher frequency in the -ve RT cells.

Figure 4: This sunburst presents a comprehensive review of the most discriminative Roget Category/Classes based on the classification model weight for a single genre, CHILDREN. We have considered top 30 discriminative features for both successful (Green) and unsuccessful books (Red).
6.2 Thematic Analysis Based on Lexical Choices

After mapping the resulting feature weights of our WordNet\textsuperscript{R$^R$} and \textsuperscript{GR} models to Roget Themes, we were able to highlight the most important Themes when classifying the success of a novel given its genre. Table 6 gives the most important themes in predicting the success of CHILDREN’S novels and the successful and unsuccessful semantic word groups within those themes. These results clearly identify words associated with "school" and "grammar" as key contributors to unsuccessful CHILDREN’S novels, while words like "secret," "enthusiastic," and "selfishness" contribute to successful CHILDREN’S novels.

The indicated Themes align with intuitive expectations for CHILDREN’S books, especially the presence of FORMATION OF IDEAS and MORAL. To verify these results, we looked at the most downloaded CHILDREN’S book, *Little Women*. We ranked each book in the CHILDREN’S genre according to the frequency of each prioritized Theme listed in Table 7. Then, we looked to see where *Little Women* ranked for each of the Themes. *Little Women*’s use of the top Themes matches up as expected, as it ranks in the top three for four of the five most important Themes, and eighth for the fifth as shown in Table 7. The opposite is true for the least downloaded books, which all rank at the bottom for use of the most important Themes.

Our Thematic observations hold true for each genre, but there is not one Theme shared by all 12 genres. This adheres to the observation we made about WNRT and \textsuperscript{GR} and each model’s lack of improvement after feature reduction for predicting success across all books independent of genre.

7 Experiments with Goodreads Rating

The discoveries made in our research are just the beginning of what can be done with our dataset. In addition to the data utilized for this project, we also extracted Goodreads Rating\textsuperscript{3} as proposed in

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|}
\hline
\textbf{Theme} & \textbf{Successful} & \textbf{Unsuccessful} \\
\hline
Affections & enthusiastic, lively, tenderness & inactive, sluggish, dull \\
Communication of Ideas & secret, untruth, language & school, grammar, taciturnity \\
Formation of Ideas & incredulity, impossibility, curiosity & dissent, sanity, memory \\
Moral & glutony, impurity, selfishness & punishment, virtue, duty \\
Personal & expecting, blemish, hopelessness & aggravation, dejection, dullness \\
\hline
\end{tabular}
\caption{Top 5 most important Themes for classifying CHILDREN novels and corresponding most predictive successful/unsuccessful thematic words}
\end{table}

Table 7: Ranking the use of the most important CHILDREN’S themes for #1 downloaded CHILDREN’S book, *Little Women* relative to other CHILDREN’S books in the dataset (Maharjan et al., 2019). We could collect the rating for 7,541 books from a total of 17,962 books scraped from Project Gutenberg, where each book has been rated by at least 5 readers. We labeled all the books having average rating $\geq 3.5$ as successful, and $< 3.5$ as unsuccessful (presented in Table 10 at Appendix A2). In Appendix A1, Table 9 demonstrates the performances of previous and our models, respectively. When predicting novel success by genre, \textsuperscript{GR} shows the best results predicting a book’s success class with an average weighted F1-score of 92.2% outperforming previous state-of-the-art results (75% for the genre attention with RNN method (Maharjan et al., 2018b)) as well. This result validates the applicability of our proposed model for book success prediction.

8 Conclusion

We created the largest dataset for evaluating book success, and presented a novel study of how context free grammar rules and semantic word association of influence a book’s success. Our empirical results demonstrate that our large dataset combined with our feature reduction technique can predict a book’s success with better accuracy than the current state-of-the-art methods. The analysis performed in this project shows the relationship between thematic word groups and a book’s popularity, with our best model that uses context free grammar lexical production rules (\textsuperscript{GR}) achieving a prediction accuracy of 94.0%. Finally, we illustrated that readers expect certain themes to be prioritized over others based on a book’s genre, and the proper use of those themes directly contributes to a book’s popularity.

\textsuperscript{3}https://www.goodreads.com/
References


Project Gutenberg. Project gutenberg. (n.d.).


Appendix

A1 Results

Table 8: Accuracy (%) of classification results BY GENRE for (Ashok et al., 2013) dataset, with/without feature reduction (R) (best performance in bold)

<table>
<thead>
<tr>
<th>MODEL (REDUCTION)</th>
<th>GENRE</th>
<th>AVG (W.F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ad</td>
<td>Ch</td>
</tr>
<tr>
<td>Unigram</td>
<td>76.0</td>
<td>71.0</td>
</tr>
<tr>
<td>UnigramR</td>
<td>92.7</td>
<td>86.0</td>
</tr>
<tr>
<td>Bigram</td>
<td>69.0</td>
<td>74.0</td>
</tr>
<tr>
<td>BigramR</td>
<td>93.8</td>
<td>92.0</td>
</tr>
<tr>
<td>POS</td>
<td>58.5</td>
<td>66.0</td>
</tr>
<tr>
<td>POSR</td>
<td>71.9</td>
<td>73.0</td>
</tr>
<tr>
<td>γU</td>
<td>64.6</td>
<td>65.0</td>
</tr>
<tr>
<td>γUR</td>
<td>100.0</td>
<td>99.0</td>
</tr>
<tr>
<td>γG</td>
<td>65.7</td>
<td>60.0</td>
</tr>
<tr>
<td>γGR</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>γR</td>
<td>56.5</td>
<td>56.0</td>
</tr>
<tr>
<td>γGR</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>γG</td>
<td>65.7</td>
<td>61.0</td>
</tr>
<tr>
<td>γGR</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>WordNet</td>
<td>74.9</td>
<td>69.0</td>
</tr>
<tr>
<td>WordNetR</td>
<td>100.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

Table 9: Average weighted F1-score for book success prediction using Goodreads rating. Part 1 of this table contains highest results of previous studies. Part 2 presents the results from the experiments with reduced feature set described in this article. Genre and model names abbreviated and the best performance is shown in bold font.

<table>
<thead>
<tr>
<th>MODEL (REDUCTION)</th>
<th>GENRE</th>
<th>AVG (W.F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ad</td>
<td>Ch</td>
</tr>
<tr>
<td>Fashok et al. (2013)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Genre Attention with RNN Maharjan et al. (2018b)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>T</td>
<td>94.0</td>
<td>89.0</td>
</tr>
<tr>
<td>γU</td>
<td>94.6</td>
<td>90.5</td>
</tr>
<tr>
<td>γGR</td>
<td>90.1</td>
<td>82.6</td>
</tr>
<tr>
<td>Roget</td>
<td>97.5</td>
<td>87.4</td>
</tr>
<tr>
<td>WN</td>
<td>79.3</td>
<td>70.3</td>
</tr>
<tr>
<td>WNRC</td>
<td>94.0</td>
<td>85.9</td>
</tr>
<tr>
<td>WNRT</td>
<td>91.5</td>
<td>77.5</td>
</tr>
<tr>
<td>TRC</td>
<td>68.7</td>
<td>60.1</td>
</tr>
<tr>
<td>T</td>
<td>91.6</td>
<td>76.6</td>
</tr>
<tr>
<td>T</td>
<td>64.1</td>
<td>59.0</td>
</tr>
<tr>
<td>γRC</td>
<td>96.2</td>
<td>80.4</td>
</tr>
<tr>
<td>γRT</td>
<td>74.7</td>
<td>57.1</td>
</tr>
</tbody>
</table>
A2 Goodreads Dataset

<table>
<thead>
<tr>
<th>GENRE</th>
<th>BOOKS</th>
<th>GR (SB)</th>
<th>GR (UB)</th>
<th>GRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adventure</td>
<td>917</td>
<td>285</td>
<td>97</td>
<td>383</td>
</tr>
<tr>
<td>Children</td>
<td>3,278</td>
<td>929</td>
<td>331</td>
<td>1,260</td>
</tr>
<tr>
<td>Detective</td>
<td>285</td>
<td>116</td>
<td>68</td>
<td>184</td>
</tr>
<tr>
<td>Drama</td>
<td>785</td>
<td>263</td>
<td>153</td>
<td>416</td>
</tr>
<tr>
<td>Fantasy</td>
<td>382</td>
<td>189</td>
<td>53</td>
<td>242</td>
</tr>
<tr>
<td>Fiction</td>
<td>5,369</td>
<td>1,461</td>
<td>722</td>
<td>2,183</td>
</tr>
<tr>
<td>Hist. Fiction</td>
<td>961</td>
<td>391</td>
<td>115</td>
<td>506</td>
</tr>
<tr>
<td>Humor</td>
<td>1,024</td>
<td>104</td>
<td>61</td>
<td>165</td>
</tr>
<tr>
<td>Poetry</td>
<td>1,664</td>
<td>441</td>
<td>140</td>
<td>581</td>
</tr>
<tr>
<td>Roma. Fiction</td>
<td>634</td>
<td>210</td>
<td>103</td>
<td>313</td>
</tr>
<tr>
<td>Sci. Fiction</td>
<td>1,748</td>
<td>388</td>
<td>581</td>
<td>969</td>
</tr>
<tr>
<td>Sho. Stories</td>
<td>915</td>
<td>214</td>
<td>125</td>
<td>339</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>17,962</strong></td>
<td><strong>4,992</strong></td>
<td><strong>2,549</strong></td>
<td><strong>7,541</strong></td>
</tr>
</tbody>
</table>

Table 10: This table presents the number of book ratings we collected from the Goodreads website for 12 genres. Here, GR stands for Goodreads, while SB, UB and GRC stands for successful books unsuccessful books and Goodreads count, respectively. We could collect a total of 7,541 book ratings from Goodreads, as opposed to the total 17,962 downloaded books from the Project Gutenberg website.
SocialVisTUM: An Interactive Visualization Toolkit for Correlated Neural Topic Models on Social Media Opinion Mining

Gerhard Hagerer\(^2\), Martin Kirchhoff\(^3\), Hannah Danner\(^2\), Robert Pesch\(^3\)
Mainak Ghosh\(^1\), Archishman Roy\(^2\), Jiaxi Zhao\(^2\), Georg Groh\(^2\)

\(^1\)Max Planck Institute for Innovation and Competition
\(^2\)Technical University of Munich
\(^3\)inovex GmbH

{gerhard.hagerer,hannah.danner,archishman.roy,jiaxi.zhao}@tum.de, grohg@mytum.de
{martin.kirchhoff,robert.pesch}@inovex.de, mainak.ghosh@ip.mpg.de

Abstract

Recent research in opinion mining proposed word embedding-based topic modeling methods that provide superior coherence compared to traditional topic modeling. In this paper, we demonstrate how these methods can be used to display correlated topic models on social media texts using SocialVisTUM, our proposed interactive visualization toolkit. It displays a graph with topics as nodes and their correlations as edges. Further details are displayed interactively to support the exploration of large text collections, e.g., representative words and sentences of topics, topic and sentiment distributions, hierarchical topic clustering, and customizable, predefined topic labels. The toolkit optimizes automatically on custom data for optimal coherence. We show a working instance of the toolkit on data crawled from English social media discussions about organic food consumption. The visualization confirms findings of a qualitative consumer research study. SocialVisTUM and its training procedures are accessible online\(^1\).

1 Introduction

Web sources, such as social networks, internet forums, and customer reviews from online shops, provide large amounts of unstructured text data. Along with the steady development of new platforms and the increasing number of internet users, the interest in methods that automatically extract the expressed opinions along with the corresponding topics and sentiments in text data has increased in recent years. Scholars and organizations from different fields can utilize such methods to identify patterns and generate new insights. Examples are opinion researchers investigating current opinions on political and societal issues, consumer researchers interested in consumers’ beliefs about the consumption and production of goods (Danner et al., 2020), and marketing managers curious about the public perception of their products and services (Berger et al., 2020; Murphy et al., 2014). (Kirchhoff, 2019)

These domain-specific use cases are of interest for research disciplines which taken by itself are not directly related to natural language processing (NLP). Consequentially, there is a constant need to provide state-of-the-art NLP methods such that domain researchers from other fields can take advantage of them. The requirements therefore are simple usage, automatic hyperparameter optimization, minimal effort for manual labeling of text data, and built-in visualizations to give an abstract overview of the discussed topics and their relation

\(^1\)https://github.com/ghagerer/SocialVisTum

---

Figure 1: SocialVisTUM applied to our use case organic food - The topics, their occurrence (in brackets) and respective correlations.
with each other.

While these practical requirements are important for domain experts, modern opinion mining approaches target specific machine learning objectives. Recently, there is a trend towards unsupervised neural methods for opinion target detection. Attention-based aspect extraction (ABAE) enables clustering of short review texts with significantly higher coherence as traditional LDA-based topic modeling, and it gives 70% F1 score for classification (He et al., 2017). This is improved recently (Karamanolakis et al., 2019; Angelidis and Lapata, 2018; Luo et al., 2019), which underlines the recent impact and potential of related techniques.

However, these have not been utilized for visualizations based on correlated topic modeling (Blei and Lafferty, 2006), where all pairs of topics “are” analyzed to determine if two topics generally tend to occur in the same texts of a given dataset. Thus, the similarity between topics can be defined. This is successfully used to connect topics (nodes) among each other based on their correlations (edges) leading to more abstract and more meaningful meta topics (graph-clusters) which additionally improves topic coherence. Consequently, these meta topics, e.g., company-related events or research sub-disciplines (Liu et al., 2014; Maiya and Rolfe, 2014), can be successfully identified by graph-based visualization techniques. However, there is a lack of related prototypes on texts discussing consumption related issues in product reviews or social media. To the best of our knowledge, there is also no related integration of sentiment analysis into a system available for potential end users, i.e., domain experts. As according text data from customers is available on a large scale in social media, this can be considered as a shortcoming in the field.

To address all denoted issues, we propose the SocialVisTUM toolkit, a new visualization and labeling tool to give users a comprehensible overview of the topics discussed in social media texts. It integrates a neural method for unsupervised sentence and comment clustering based on word vectors and attention. We denote the respective clusters as topics hereafter. In addition, we provide a graph-based visualization showing the topics as labeled nodes and the correlation between them as edges. A force-directed graph layout maintains readability even while many relevant topics and topic relations are displayed. (Kirchhoff, 2019)

In our interactive graphical user interface, the number of topics displayed and the correlation threshold required to display a connection between two topics can be dynamically adjusted. Further, contextual topic information is provided, such as the number of respective topic occurrences in the social media texts as node diameter, the correlation between the topic occurrences as edge width, example sentences from the data for each topic, a list of representative words for each topic, and the regarding sentiment distribution of a topic. It is a common practice to represent topics merely by word lists (Blei et al., 2003; Chen et al., 2014), which tend to be insufficient to comprehensively express a topic on our given dataset. (Kirchhoff, 2019)

To avoid manual labeling and to give users an immediate impression of each topic, topic labels are generated automatically based on a custom algorithm utilizing the most common WordNet hypernym in a topic’s top words. Furthermore, we find that topic hypernym statistics can serve as a metric for automatic hyperparameter optimization, which in our case gives practical advantages over widely used coherence scoring metrics.

In addition to a more detailed description of our SocialVisTUM toolkit, we show the results of a case study based on social media texts from online commenters debating about organic food consumption. We demonstrate that the correlated topics give a meaningful graph representation of the social media discussions supporting the understanding of the concerns of consumers. In this regard, we also show how the combined illustration of different types of relevant topic and sentiment information and automatic labeling of clusters are a contribution.

2 Related Work

Correlated topic models were introduced 2006 (Blei and Lafferty, 2006; Li and McCallum, 2006) to improve topic coherence and to provide graph visualizations based on topics as nodes and their correlations as edges. This shows potential to improve text mining for the end user as “powerful means of exploring, characterizing, and summarizing large collections of unstructured text documents” (Maiya and Rolfe, 2014). Meta topics, such as research domains and their inter-disciplinary overlaps, can thus be described clearly, automatically, and empirically (Blei and Lafferty, 2007).

These correlated topic models are applied for
more sophisticated visualization approaches. TopicPanorama models technology-related topics from various text corpora, including newspaper articles, blogs, and tweets (Liu et al., 2014). Here, the domain expert is given the option to interactively modify the matching result of the labeled topic graph. Another topic visualization called topic similarity networks is particularly addressing the visualization of large document sets (Maiya and Rolfe, 2014). While claiming good scalability regarding the number of documents, beneficial methods to achieve automatic topic labeling are successfully quantified. TopicAtlas provides a graphical user interface to explore text networks, such as hyperlinked webpages and academic citation networks. For manual mining purposes, topic models are generated and related to one another to facilitate manual navigation and finding of relevant documents (He et al., 2016). These examples show a steady, meaningful, and promising development regarding the visualization of correlated topic modeling, partially also applied to social media texts such as micro-blogs. However, these examples do not include sentiment analysis as means to conduct market research and quantify customer satisfaction in specific and not yet explored market domains. Furthermore, the widely used latent Dirichlet allocation (LDA) technique tends to be incoherent on short texts, such as, product reviews or social media comments, and thus insufficient to detect opinion targets in an unsupervised manner (He et al., 2017).

Automatic topic coherence optimization can be seen as desirable for a topic modeling visualization toolkit such as SocialVisTUM, which tries to minimize manual optimization efforts for non-technical users. Therefore, we refer to two widely used coherence definitions (Ghosh, 2020). Firstly, word co-occurrence-based methods measure how often pairs of representative topic words co-occur in the training data set or in an external reference data set. In that regard, it has been shown that the evaluation methods UMass, UCI and NPMI correlate with human judges (Stevens et al., 2012; Newman et al., 2010; Mimno et al., 2011; Bouma, 2009; Ding et al., 2018) and are considered to be a default metric for topic coherence. Secondly, word embedding similarity based coherence scores are recently utilized as these are also based on word co-occurrence statistics (Pennington et al., 2014) and behave similar to NPMI coherence scoring (Ding et al., 2018), resulting in high correlation with human perception. These methodologies show the undesirable effect of no distinct local optimum when the hyperparameters of the models are changed, e.g., number of clusters or vocabulary size. On our data, these parameters increase together with the coherence scores, while the subjective performance, i.e., the human perception, actually decreases. We describe this effect and our solution in the case study section.

3 Clustering Architecture

The unsupervised neural network model called attention-based aspect extraction (ABAE) (He et al., 2017) clusters sentences based on GloVe word embeddings (Pennington et al., 2014) and attention (Bahdanau et al., 2014) to focus on the most important words in a sentence. Every sentence \( s \) is represented by a vector \( z_s \) that is defined as the weighted average of all the word vectors of that sentence. The weights are attentions calculated based on the contribution of the respective words to the meaning of the sentence and the relevance to the topics. These topics are defined by the actual centroids. In their publication, the topics are mapped to aspect classes for unsupervised aspect extraction, which we do not do for our case. (Kirchhoff, 2019)

The topics are initialized as the resulting centroids of k-means clustering on the word embeddings of the corpus dictionary. These are then stacked as topic embedding matrix \( T \). During training, ABAE calculates sentence reconstructions \( r_s \) for each sentence. These are linear combinations of the topic embeddings from \( T \) and defined as

\[
r_s = T^T \cdot p_t, \tag{1}
\]

where \( p_t \) is the weight vector over \( K \) topic embeddings. Each weight corresponds to the probability that the input sentence belongs to the associated topic. \( p_t \) is obtained by reducing the dimension of \( z_s \) to the number of topics \( K \) and applying softmax such that

\[
p_t = \text{softmax} (W \cdot z_s + b), \tag{2}
\]

where \( W \) and \( b \) are trainable and matrix weights and a bias vector respectively. The topic embeddings \( T \) are updated during training to minimize the reconstruction error \( J(\theta) \) between \( r_s \) and \( z_s \) based on the contrastive max-margin objective function. Since words and topics share the same dimensionality, cosine similarity between both can be used to
look up the most similar words representing each topic, similar to the way LDA (Blei et al., 2003) represents topics as word distributions. (Kirchhoff, 2019)

4 The SocialVisTUM Toolkit

Visualization Figure 2 shows an example of the visualization and labeling tool. Topics are represented as nodes with according labels, and the number of texts assigned to the topics is given in parentheses next to the label. The node size increases based on the number of topic occurrences. The edges of the topic connections are labeled by the topic correlations. The link thickness increases with a higher positive correlation. A graph layout based on repelling forces between nodes helps to avoid overlaps, which is especially useful when many nodes and links are displayed. A second force keeps the graph centered. Users can also move nodes around to get a more comprehensible overview. (Kirchhoff, 2019)

Topic Nodes and Correlations After training the ABAE model, the sentences are assigned to topics based on the maximum topic probability from $p_t$, see formula 2. The correlation between two topics $i$ and $j$ is calculated based on the probabilities $(p_{ti})_i$ and $(p_{tj})_j$ of each given sentence $t$. This yields a value in the range of $[-1, 1]$ for every pair of topics specifying the strength of the corresponding relatedness. (Kirchhoff, 2019)

Hiding Insignificant Topics An occurrence threshold slider defines the percentage of sentences that must be about a topic to display the associated node. Another slider can be used to set the correlation threshold to define the required positive or negative correlation to display the associated connections. These sliders are especially helpful to maintain a clear visualization by limiting the number of shown topics and connections when there are many of them available.

Topic Inspection Users can double-click a node to receive additional information about a topic, i.e., representative words and sentences, as shown in Figure 5 on the left and right respectively. As representative words, the top 10 words are shown sorted by the distance of their embeddings to the selected topic centroid in ascending order. The representative sentences are the ones with the highest probability from $p_t$ for the given topic. During topic inspection mode, only nodes that are connected to the clicked node and the associated links are displayed. A double click on the same node brings back the whole graph again.

Colorization of Topic Nodes In an updated version of SocialVisTUM, we introduce two meaningful colorings of the topic nodes for correlated topic clustering and sentiment analysis.

Firstly, we perform a hierarchical clustering algorithm such that those topics which are strongly correlated with each other are colorized in one and the same color respectively. A dynamic slider GUI element helps to adjust the correlation threshold accordingly. One example outcome is shown in 2.

Secondly, we perform sentiment analysis using the Valence Aware Dictionary and sEntiment Reasoner or VADER method (Gilbert, 2014). It is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media settings. It gives positive, negative, and neutral scores to represent the proportion of text in that sentence that falls in these categories. For each sentence, we use the compound score, i.e., the sum of all lexicon ratings normalized between -1 (most negative) and +1 (most positive). We then calculate the average sentiment score for each topic based on all respective topic sentences. In Figure 3, positive sentiment is shown as green topic nodes, and negative as red. (Roy and Zhao, 2020)
Table 1: Example topics, automatically assigned topic labels, and representative words. The value next to the topic label denotes how often the label occurs as a shared hypernym. The number of hypernyms on the right tells in how many word comparisons a shared hypernym is identified. Taken from (Kirchhoff, 2019).

<table>
<thead>
<tr>
<th>Topic Label</th>
<th>Representative words</th>
<th># Hypernyms</th>
</tr>
</thead>
<tbody>
<tr>
<td>animal (102)</td>
<td>insect, ant, habitat, rodent, herbivore</td>
<td>218</td>
</tr>
<tr>
<td>compound (91)</td>
<td>amino, enzyme, metabolism, potassium, molecule</td>
<td>158</td>
</tr>
<tr>
<td>chemical (74)</td>
<td>fungicide, insecticide, weedkiller, preservative, bpa</td>
<td>131</td>
</tr>
<tr>
<td>systematically (0)</td>
<td>systematically, adequately, cleaned, properly, milked</td>
<td>0</td>
</tr>
</tbody>
</table>

Automatic Topic Labels We introduce an approach to label topic nodes automatically. It is based on shared hypernyms, i.e., the lowest common denominator for words, which we identify using the representative topic words denoted in the previous paragraph and the lexical database WordNet (Miller, 1995). First, we retrieve the hypernym hierarchy for every representative topic word, as shown in Figure 4, and compare every word with every other word in the word list. Next, at each comparison, we save the hypernym with the lowest distance to the compared words in the hypernym hierarchy. We denote these as shared hypernyms. We only consider hypernyms if their distance to the word is smaller than half of the distance to the root hypernym to avoid unspecific labels like entity and abstraction. Eventually, we employ the hypernym that occurs most often as topic labels. If no hypernym can be identified, we use the most representative word instead. In the example shown in figure 4, we identify dairy product as the lowest shared hypernym of yoghurt and butter, and food as lowest shared hypernym of yoghurt and bread. (Kirchhoff, 2019)

The quality of a shared hypernym chosen as topic label can be approximated by inspecting the number of its hypernym occurrences – see table 1. Topic labels occurring frequently as shared hypernym are usually suitable (e.g., animal (102) and compound (91)) in contrast to topic labels that rarely occur (e.g. group_action (9) or smuckers (0)). Thus, we conclude that the number of hypernym occurrences of each topic is suitable to estimate the topic coherence for hyperparameter optimization – see section 5.2 later on. (Kirchhoff, 2019)

Changing Labels To change the label of a topic, the user can click on the associated label of a node. This opens a prompt allowing the user to change the topic label. The user can download a JSON file with the updated labels by clicking on the Create file button on the sidebar. (Kirchhoff, 2019)

5 Case Study

We demonstrate SocialVisTUM’s potential for social media data exploration on a new data set.

5.1 Data Set

We crawled online user comments on organic food from multiple forums (e.g., Reddit, Quora, Disqus) and the comment sections of news websites (The Washington Post, The New York Times, Chicago Tribune, HuffPost, and many more). The goal is to discover the discussed topics and opinions in social media regarding the organic food consumption.

Relevant articles from the platforms are found by the search terms “organic food”, “organic agriculture”, and “organic farming”. We further filtered for domain relevance by applying naive Bayes classification on bag of words trained on 1000 random and accordingly labeled texts (84.70% accuracy with 10-fold cross validation). From the left texts, we retain comments containing either of the words food and organic. The left data set consists of 515,347 sentences totaling 83,938 posts, which are used to train the ABAE topic model. We use the 300-dimensional pre-trained GloVe embeddings and fine-tuned them on the data. (Kirchhoff, 2019)

5.2 Hyperparameter Estimation

Some hyperparameters of the utilized ABAE topic model are the number of topic clusters and the vocabulary size. To optimize these automatically, we define a new metric, the average number of shared hypernyms (ANH). We first derive the frequency of all shared hypernyms for each topic, which is already done for automatic topic labeling in section 4. The ANH is the sum of hypernym frequencies over all topics divided by the number of topics. (Kirchhoff, 2019)
In our case study, we identified the following advantages of ANH over the widely used coherence score (CS). First, an increasing number of topics does not always increase the ANH, as a high number of topics leads to many incoherent topics with fewer shared hypernyms, i.e., a lower ANH. Second, a medium-sized vocabulary (~10,000 words) produces the most coherent topics according to ANH and manual inspection. Table 2 shows an excerpt of the results for varying parameters.

5.3 Interpretation

We applied SocialVisTUM to our case of organic food yielding the topics displayed in figure 1. A consumer researcher in the domain of organic food manually refined the automatic labeling based on the most similar words of each topic. The topics reflect previous findings of a qualitative content analysis on a small sub-sample of our data set (Danner and Menapace, 2020). The correlated topics allow market researchers to investigate the context in which topics are discussed.

Figure 5 takes a closer look at the example topic *pesticides*, which is concerned with different pesticides and their toxicity. The topic *pesticides* is correlated with the topic *organic_production_standards*, which references different organic or related production methods, such as bio-dynamic, hydroponic, or bio-intensive agriculture. This correlation suggests that, for the commenting users in our data set, the non-use of chemical-synthetic pesticides is an important characteristic of organic compared to non-organic production. Further topics correlated with *pesticides* propose that the commenting users are concerned about the use of *pesticides* in farming and that they discuss the issue of *pesticides*, possibly the residues thereof, in the context of different *food_products*.

6 Conclusion

In this paper, a case of the proposed SocialVisTUM demonstrates the visualization of coherent topics on a given corpus of social media texts about organic food. The graph-based visualization with topics as nodes and topic correlations as edges reflects the topics and patterns found in a related qualitative content analysis (Danner and Menapace, 2020). The presentation of additional topic information, such as word lists, representative sentences, topic importance, and meaningful predefined labels, provide a basis for the understanding and interpretation of a topic for domain experts. The integrated hyperparameter optimization automatically yields interpretable topics and helps tailoring the model to the given data set. For future work, we plan to evaluate the correlated topics on other corpora and in other use cases. In addition to Pearson correlation, other correlations could improve the approach. We plan to integrate multi-lingual word features, such as BERT (Devlin et al., 2018), for cross-cultural comparisons.

Acknowledgments

We thank inovex GmbH for supporting the research of this paper by providing computational resources, Robert Pesch and Martin Kirchhoff for their important contributions. Paragraphs adopted from student works are followed by the corresponding citation after the punctuation.
References

Stefanos Angelidis and Mirella Lapata. 2018. Summarizing opinions: Aspect extraction meets sentiment prediction and they are both weakly supervised. CoRR, abs/1808.08858.


Ling Luo, Xiang Ao, Yan Song, Jin Yao Li, Xiaopeng Yang, Qing He, and Dong Yu. 2019. Unsupervised neural aspect extraction with sememes. In IJCAI, pages 5123–5129. ijcai.org.


486


### Appendix A. Comparison of Coherence Metrics

<table>
<thead>
<tr>
<th># Topics</th>
<th>Voc. Size</th>
<th>CS</th>
<th>ANH</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1,000</td>
<td>-1104</td>
<td>28.6</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>-765</td>
<td><strong>68.0</strong></td>
</tr>
<tr>
<td></td>
<td>18,000</td>
<td>-403</td>
<td>5.2</td>
</tr>
<tr>
<td>15</td>
<td>1,000</td>
<td>-366</td>
<td>33.3</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>-270</td>
<td><strong>40.0</strong></td>
</tr>
<tr>
<td></td>
<td>18,000</td>
<td>-197</td>
<td>33.8</td>
</tr>
<tr>
<td>50</td>
<td>1,000</td>
<td>-110</td>
<td>30.4</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>-70</td>
<td><strong>51.8</strong></td>
</tr>
<tr>
<td></td>
<td>18,000</td>
<td>-54</td>
<td>49.7</td>
</tr>
</tbody>
</table>

Table 2: Comparing two coherence metrics: coherence score (CS) and average number of shared hypernyms (ANH). The advantage of ANH is that it has its global optimum always in the middle as opposed to CS. This property is beneficial for hyperparameter optimization.
Apples to Apples: A Systematic Evaluation of Topic Models

Ismail Harrando, Pasquale Lisena and Raphaël Troncy
EURECOM, Sophia Antipolis, France
{firstname.lastname}@eurecom.fr

Abstract
From statistical to neural models, a wide variety of topic modelling algorithms have been proposed in the literature. However, because of the diversity of datasets and metrics, there have not been many efforts to systematically compare their performance on the same benchmarks and under the same conditions. In this paper, we present a selection of 9 topic modelling techniques from the state of the art reflecting a diversity of approaches to the task, an overview of the different metrics used to compare their performance, and the challenges of conducting such a comparison. We empirically evaluate the performance of these models on different settings reflecting a variety of real-life conditions in terms of dataset size, number of topics, and distribution of topics, following identical preprocessing and evaluation processes. Using both metrics that rely on the intrinsic characteristics of the dataset (different coherence metrics), as well as external knowledge (word embeddings and ground-truth topic labels), our experiments reveal several shortcomings regarding the common practices in topic models evaluation.

1 Introduction
The automatic analysis of textual data has gained increasing levels of attention over the last few decades. The cost of manually analysing and annotating the ever-growing quantity of content created and shared on the Web continues to be prohibitively expensive. Topic modelling is an NLP task where, given a corpus of documents, the objective is to find the underlying meaningful clusters of documents (or topics) that are thematically coherent (use consistent and related vocabulary) and assign each document to one or more of these topics. As a text mining technique, it allows the analysis of big volumes of textual documents through clustering them into coherent sets addressing similar subjects (or topics), and labeling them using keywords that are understandable by the end-user. It has the advantage of not relying on any labeled data to achieve good results, as the training of topic models is done in an unsupervised matter. Moreover, the resulting topics and representations can then be used to perform other NLP tasks such as trend prediction (Lau et al., 2012), text summarization (Lin and Hovy, 2000), improving named entity recognition (Newman et al., 2006), and content recommendation (Papneja et al., 2021).

Because of the unsupervised nature of the task, the evaluation of the quality of topic modelling techniques relies usually on metrics that do not require human annotation or ground-truth labels. Most of the used “coherence” metrics – further detailed in Section 3.1 – attempt to measure how much the resulting topics reflect some statistical characteristics of the original dataset and its word co-occurrences distribution. These metrics utilise different definitions of what a “coherent topic” is, and they only contingently agree with humans judgement (Chang et al., 2009). Coupled with the different approaches for document preprocessing and the variety of used evaluation datasets, this complexity leads to several nuances in the evaluation process that are not widely acknowledged in the literature at large. Thus, comparisons can be inconsistent and sometimes misleading.

In this work, we selected a diverse array of topic modelling algorithms (probabilistic, algebraic, embedding-based and neural) from the literature and we provide a thorough comparison using a unified evaluation protocol. This protocol evaluates each topic model on several datasets, using a variety of metrics that range from intrinsic evaluation of the clustering quality to ones that assess the alignment between the extracted topics and the human-assigned labels. With this strategy, we aim to illustrate the inconsistency of these metrics when
varying several subtle evaluation conditions. We analyse the results and we discuss the differences in performances across the different algorithms, datasets and parameters.

The remainder of this paper is organised as follows. In Section 2, we describe some related work, detailing some state-of-the-art topic modelling techniques. Different metrics for evaluating topic models are introduced in Section 3, while Section 4 describes the datasets we use for this purpose. In Section 5, we extensively analyse 9 topic models using coherence and ground truth related metrics. Finally, we provide some conclusions in Section 6.

2 Related Work

2.1 Topic Modelling Techniques

One of the first yet still widely used techniques is Latent Dirichlet Allocation (LDA) (Blei et al., 2003), an unsupervised statistical modelling approach that considers each document as a bag of words and creates a randomly assigned document-topic and word-topic distributions. Iterating over words in each document, the distributions are updated according to the probability that a document or a word belongs to a certain topic. The Hierarchical Dirichlet Process (HDP) model (Teh et al., 2006) considers instead each document as a group of words belonging with a certain probability to one or multiple components of a mixture model, i.e. the topics. Both the probability measure for each document (distribution over the topics) and the base probability measure – which allows the sharing of clusters across documents – are drawn from Dirichlet Processes (Ferguson, 1973). Unlike most other topic models, HDP infers the number of topics automatically. Gibbs Sampling for a DMM (GSDMM) applies the Dirichlet Multinomial Mixture model for short text clustering (Yin and Wang, 2014). This algorithm works by computing iteratively the probability that a document join a specific one of the N available clusters. This probability consists of two parts: 1) a part that promotes the clusters with more documents; 2) a part that advantages the movement of a document towards similar clusters, i.e. which contains a similar word-set.

Recently, pre-trained Word vectors such as word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014) have been used to help to enhance topic-word representations, as achieved by the Latent Feature Topic Models (LFTM) (Nguyen et al., 2015). One of the LFTM algorithms is Latent Feature LDA (LF-LDA), which extends the original LDA algorithm by enriching the topic-word distribution with a latent feature component composed of pre-trained word vectors. In the same vein, the Paragraph Vector Topic Model (PVTM) (Lenz and Winker, 2020) uses doc2vec (Le and Mikolov, 2014) to generate document-level representations in a common embedding space. Then, it fits a Gaussian Mixture Model to cluster all the similar documents into a predetermined number of topics – i.e. the number of GMM components.

Topic modelling can also be performed via linear algebraic methods. Starting from the high-dimensional term-document matrix, multiple approaches can be used to lower its dimensions. Then, we consider every dimension in the lower-rank matrix as a latent topic. A straightforward application of this principle is the Latent Semantic Indexing model (LSI) (Deerwester et al., 1990), which uses Singular Value Decomposition as a means to approximate the term-document matrix (potentially mediated by TF-IDF) into one with fewer rows – each one representing a latent semantic dimension in the data – and preserving the similarity structure among columns (terms). Non-negative Matrix Factorisation (NMF) (Paatero and Tapper, 1994) exploits the fact that the term-document matrix is non-negative, thus producing not only a denser representation of the term-document distribution through the matrix factorisation but guaranteeing that the membership of a document to each topic is represented by a positive coefficient.

In recent years, neural network approaches for topic modelling have gained popularity giving birth to a family of Neural Topic Models (NTM) (Cao et al., 2015). Among those, doc2topic (D2T) uses a neural network which separately computes N-dimensional embedding vectors for words and documents (with N = number of topics) before computing the final output using a sigmoid activation. The distributions topic-word and document-topic are obtained by getting the final weights on the two embedding layers. The Contextualized Topic Model (CTM) (Bianchi et al., 2020) uses Sentence-BERT (SBERT) (Reimers and Gurevych, 2019) – a neural transformer language model designed to compute sentences representations efficiently – to generate
a fixed-size embedding for each document to contextualise the usual Bag of Words representation. CTM enhances the Neural-ProdLDA (Srivastava and Sutton, 2017) architecture with this contextual representation to significantly improve the coherence of the generated topics.

2.2 Topic Models Comparison

To the best of our knowledge, no extensive comparison of recent topic models – covering multiple metrics and datasets under the same preprocessing condition – has been made. Some previous works have tried to compare different topic models on certain datasets and metrics. A review of statistical topic modelling techniques is included in Newman et al. (2006). Schofield and Mimno (2016) provide a comparison resulting from the effect of preprocessing on the performance of LDA on multiple corpora. Jelodar et al. (2017) offer a survey of topic modelling techniques based on LDA, as well as their different applications in recent literature. Yi and Allan (2009) and Alexander and Gleicher (2016) compare several topic models, evaluated as tools for performing Information Retrieval downstream tasks such as Topic Alignment, Change Comparison, Document Retrieval and Query Expansion. Several evaluation metrics based on top-words analysis was suggested by Newman et al. (2010). Alghamdi and Alfalqi (2015) compare 4 topic models (LDA, LSI, PLSA and CTM): this survey studied both their capability in modelling static topics, as well as in detecting topic change over time, highlighting the strengths and weaknesses of each. Burkhardt and Kramer (2019) provide a survey for the adjacent task of multi-label topic models, underlining its challenges and promising directions. Qiang et al. (2020) give an extensive performance evaluation of multiple topic models in the context of the Short Text Topic modelling sub-task (e.g. tweets). Finally, Doogan and Buntine (2021) studied several topic model coherence measures to assess how informative they are in several applied settings revolved around interpretability as an objective. They showed how standard coherence measures may not inform the most appropriate topic model or the optimal number of topics when measured up against human evaluation, thus challenging their utility as quality metrics in the absence of ground truth data.

2.3 Metrics

While our work utilises multiple comparison metrics (detailed in Section 3.1), it is worth highlighting that many other evaluation metrics were proposed in the literature to expose different characteristics of the studied topic models such as Classification Accuracy and Perplexity (Qiang et al., 2020), Entropy and Held-out Likelihood (Schofield and Mimno, 2016), Stability (Alexander and Gleicher, 2016), and Top-word Ranking (Greene et al., 2014), whereas finding a universally useful metric for topic modelling evaluation is still an open problem (Blei, 2012; Doogan and Buntine, 2021; Hoyle et al., 2021).

3 Metrics

The evaluation of machine learning techniques often relies on accuracy scores computed comparing predicted results against a ground truth. In the case of unsupervised techniques like topic modelling, the ground truth is not always available. For this reason, in the literature, we can find:

- metrics which enable to evaluate a topic model independently from a ground-truth, among which, coherence measures are the most popular ones (Röder et al., 2015; O’Callaghan et al., 2015; Qiang et al., 2020);
- metrics that measure the quality of a model’s predictions by comparing its resulting clusters against ground truth labels, in this case a topic label for each document.

3.1 Coherence Metrics

The coherence metrics rely on the joint probability $P(w_i, w_j)$ of two words $w_i$ and $w_j$ that is computed by counting the number of documents in which those words occur together divided by the total number of documents in the corpus. The documents are fragmented using sliding windows of a given length, and the probability is given by the number of fragments including both $w_i$ and $w_j$ divided by the total number of fragments. This probability can be expressed through the Pointwise Mutual Information (PMI), defined as:

$$PMI(w_i, w_j) = \log \frac{P(w_i, w_j) + \epsilon}{P(w_i) \cdot P(w_j)}$$ (1)

A small value is chosen for $\epsilon$, in order to avoid computing the logarithm of 0. Different metrics based on PMI have been introduced in the literature,
differing in the strategies applied for token segmentation, probability estimation, confirmation measure, and aggregation. The UCI coherence (Röder et al., 2015) averages the PMI computed between pairs of topics, according to:

$$C_{UCI} = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} PMI(w_i, w_j) \quad (2)$$

The UMASS coherence (Röder et al., 2015) relies instead on a different joint probability:

$$C_{UMASS} = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \log \frac{P(w_i, w_j)+\epsilon}{\rho(w_i, w_j)} \quad (3)$$

The Normalized Pointwise Mutual Information (NPMI) (Chicarcos et al., 2009) applies the PMI in a confirmation measure for defining the association between two words:

$$NPMI(w_i, w_j) = \frac{PMI(w_i, w_j)}{-\log(P(w_i, w_j)+\epsilon)} \quad (4)$$

NPMI values go from -1 (never co-occurring words) to +1 (always co-occurring), while the value of 0 suggests complete independence. The most common implementation of $C_{NPMI}$ applies NPMI as in Eqn (4) to couples of words, computing their joint probabilities using sliding windows.

This measure can be applied also to word sets. This is made possible using a vector representation in which each feature consists in the NPMI computed between $w_i$ and a word in the corpus $W$, according to the formula:

$$\hat{v}(w_i) = \{NPMI(w_i, w_j) | w_j \in W\} \quad (5)$$

The vectors related to each word of the topic are then compared using the cosine similarity $C_V$.

Fang et al. (2016) introduce Word Embeddings-based Coherence. This metric relies on pre-trained word embeddings such as GloVe or word2vec and evaluates the topic quality using a similarity metric between its top words. In other words, a high mutual embedding similarity between a model’s top words reflects its underlying semantic coherence. In this paper, we will use the sum of mutual cosine similarity computed on the GloVe vectors\(^2\) of the top 10 words of each topic.

$$C_{WE} = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \cos(v_i, v_j) \quad (6)$$

where $v_i$ and $v_j$ are the GloVe vectors of the words $w_i$ and $w_j$.

In practice, these metrics are computed at the topic level and then aggregated using the arithmetic mean, in order to provide a coherence value for the whole model.

### 3.2 Metrics Which Relies on a Ground Truth

The most used metric that relies on a ground truth is the Purity, defined as the fraction of documents in each cluster with a correct prediction (Hajjem and Latiri, 2017). A prediction is considered correct if the original label coincides with the original label of the majority of documents falling in the same topic prediction. Given $L$ the set of original labels and $T$ the set of predictions:

$$Purity(T, L) = \frac{1}{|T|} \sum_{i \in T} \max_{j \in L} |T_i \cap L_j| \quad (7)$$

Other metrics are used in the literature for evaluating the quality of classification or clustering algorithms, applied to the topic modelling task:

1. Homogeneity: a topic model output is considered homogeneous if all documents assigned to each topic belong to the same ground-truth label (Rosenberg and Hirschberg, 2007);

2. Completeness: a topic model output is considered complete if all documents from one ground-truth label fall into the same topic (Rosenberg and Hirschberg, 2007);

3. V-Measure: the harmonic mean of Homogeneity and Completeness. A V-Measure of 1.0 corresponds to a perfect alignment between topic model outputs and ground truth labels (Rosenberg and Hirschberg, 2007);

4. Normalized Mutual Information (NMI) is the ratio between the mutual information between two distributions – in our case, the prediction set and the ground truth – normalised through an aggregation of those distributions’ entropies (Lancichinetti et al., 2009). The aggregation can be realised by selecting the minimum/maximum or applying the geometric/arithmetic mean. In the case of arithmetic mean, NMI is equivalent to the V-Measure.

In this work, we use their implementations as provided by scikit-learn (Pedregosa et al., 2011).
4 Datasets

In this section, we introduce the datasets that we use in our experiments. The features of each dataset are reported in Table 1.

A common pre-processing is performed on the datasets before training, consisting of:

- Removing numbers, which, in general, do not contribute to the broad semantics of the document;
- Removing the punctuation and lower-casing the text;
- Removing the standard English stop words;
- Lemmatisation using Wordnet, to deal with inflected forms as they are a single semantic item;
- Ignoring words with 2 letters or less. In facts, they are mainly residuals from removing punctuation – e.g. stripping punctuation from people’s produces people and s.

The same pre-processing is also applied to the text before topic prediction.

4.1 20 NewsGroups

The 20 NewsGroups collection (20NG) (Lang, 1995) is a popular dataset used for text classification and clustering. It is composed of English news documents, distributed fairly equally across 20 different categories according to the subject of the text. We use a reduced version of this dataset, which excludes all the documents composed by the sole header while preserving an even partition over the 20 categories. This reduced dataset contains 11,314 documents. We pre-process the dataset to remove irrelevant metadata – consisting of email addresses and news feed identifiers – keeping just the textual content.

4.2 Agence France Presse

The Agence France Presse (AFP) publishes daily up to 2000 news articles in 5 different languages, together with some metadata represented in the NewsML XML-based format. Each document is categorised using one or more subject codes, taken from the IPTC NewsCode Concept vocabulary. In the case of multiple subjects, they are ordered by relevance. In this work, we only consider the first level of the hierarchy of the IPTC subject codes.

We extracted a subset containing 125,516 news documents in English released in 2019.

4.3 Yahoo! Answers Comprehensive Q&A

The Yahoo! Answers Comprehensive Q&A (later simply Yahoo) contains over 4 million questions and their answers, as extracted from the Yahoo! Answers website. Each question comes with metadata such as title, date, and category, as well as a list of user-submitted answers. We construct documents by concatenating the title, body and best answer for each question – following Zhang et al. (2015) – and preprocess the documents in the same way as mentioned above. Then we create 2 subsets:

- **Yahoo balanced**, in which each category is represented by the same number of documents (1000) for a total of 26,000 documents;
- **Yahoo unbalanced**, in which the number of documents sampled from each category is proportional to its presence in the overall dataset, for a total of 22,121 documents.

These two subsets have been realised having a number of documents of the same order of magnitude. This allows to compare the differences in performance with balanced and unbalanced sets.

Table 1 summarises the properties of these datasets. The datasets present multiple differences, namely the size, the length of the documents and the distribution of documents per topic (i.e. ground truth label).

5 Experiment and Results

Evaluating an unsupervised task such as Topic Modelling is inherently challenging, and despite the variety of metrics, it is still an open problem (Hoyle et al., 2021). While intrinsic metrics (coherence) try to measure the underlying quality of the topical clusters generated by each model, they do not always match with human judgement. Two very coherent topics (according to the metric) can still fall under the same topic label for a human, and vice-versa. Topic models aim to maximise the posterior probability of a document belonging to a coherent topic, regardless of how it maps to human-perceived categories. For instance, Christianity and Atheism can be both filed as two independent topics or one topic (religion) by a human annotator, and while neither arbitrary option is wrong, it constitutes a big difference to how we would evaluate the topic modelling algorithms. They have no means

---

3https://github.com/selva86/datasets/
4http://medialab.afp.com/afp4w/
5http://cv.iptc.org/newscodes/subjectcode/
6https://answers.yahoo.com
of inferring what humans find to be *topically distinct* beyond co-occurrence statistics, making the comparison to human-annotated labels (as a “gold standard”) quite insufficient. Because of these challenges, few works in the literature (O’Callaghan et al., 2015; Alexander and Gleicher, 2016; Alghamdi and Alfalqi, 2015; Qiang et al., 2020) go beyond simple comparisons that only use one metric or dataset, eclipsing merits and shortcomings of the other methods. We attempt to provide a more thorough comparison using multiple evaluation datasets – varying in size, document length, number of topics, and label distribution – and metrics from the literature as a step towards a better understanding of the available options and their usability for different potential use-cases.

### 5.1 Varying the datasets

This section reports a comparison between 9 topic modelling algorithms described in Section 2. Our experimental setup goes as follows:

- For each dataset, we pre-process every document using the process described in Section 4;
- We train each topic model on each dataset, selecting the hyper-parameters through an optimisation process based on grid search, in order to maximise the $C_{NPMI}$ score. The use of a coherence metric as an optimisation objective is justified by the common use-case scenario, in which ground-truth labels are not present. The full set of parameters is documented in the repository\(^7\);  
- For each trained model, we compute all the intrinsic (coherence) metrics and the ground-truth-based ones.

For the experiment, we rely on To-ModAPI (Lisena et al., 2020), an open-source topic modelling API that is built to easily train, evaluate and compare several topic models. This framework provides a common interface for training, performing topic inference, and evaluating using coherence and ground truth. It includes all the metrics described above.

The number of topics – which must be provided in input to the algorithm for training – has been set to 20, 17 and 26 respectively when training on 20NG, AFP, and Yahoo, to mimic the original number of labels in each corpus. HDP has not been concerned with the choice of the number of topics, because it automatically infers it. For the first two datasets, we perform another training using the same hyper-parameters but increasing the number of topics to 50, to study its effect on the performance on the various metrics.

While all the obtained results are available in the appendix\(^8\), we will report in Figure 1 a selection of the most noticeable scores, namely $C_{NPMI}$, Word Embeddings coherence and V-Measure.

$C_{NPMI}$ values are in line with all the other coherence metrics in terms of ranking (listed in the appendix for brevity), i.e. LDA shows consistently good coherence scores across all datasets, followed by NMF and PVTM.

For the CTM model, we obtained a significantly lower coherence value than the one reported by Bianchi et al. (2020). Further investigation and experiments revealed the impact of an additional preprocessing step which reduces the vocabulary to the 2000 most frequent words. This further preprocessing improves the NPMI score of CTM from $-0.028$ to 0.116, while lowering the one of LDA from 0.133 to 0.126. This confirms the limits of topic modelling comparison and enforces the call for a standard procedure.

Word embeddings coherence demonstrated a better correlation with human judgement (Fang et al., 2016). Unsurprisingly, the two models that rely on word embeddings (LFTM, PVTM) tend to perform notably better (Figure 1).

The V-measure results included in Figure 1 are particularly relevant for understanding the correlation between the predicted topics and the ground-truth labels.}

\(^7\)https://github.com/D2KLab/ToModAPI/blob/master/params.md  
\(^8\)https://github.com/D2KLab/ToModAPI/blob/master/appendix.pdf
Figure 1: NPMI, Word embedding coherence and V-measure across the models trained on the different datasets.

Figure 2: NPMI of each model on the 20NG dataset when varying the number of topics.
truth, as it summarises three metrics – homogeneity, completeness and purity. This metric relies on human choices – made either by the editors for AFP or the website users for 20NG and Yahoo – and so it approximates the correlation between the topics as decided by the algorithms and the human (subjective) judgement on the same matter. Again, LDA is leading in overall performances, while other models – LFTM, PVTM, GSDMM – have good scores on particular datasets. The Yahoo dataset is particularly challenging for all models (the maximum value for V-measure is 0.33 for LDA), as compared to AFP (0.55 for LDA) or 20NG (0.59 for PVTM). This is probably due to a combination of document length, noise and errors in user-submitted content, and the potential overlap in topics. Increasing the number of topics systematically improves the results on AFP, raising the Homogeneity and Purity scores. This happens because the more a topic is granular, the highest is the chance that it maps correctly to the human label is correct. However, this is not observed on 20NG. Given the difference in size between 20NG and AFP, we conclude that the dimension of the former is not allowing it to extract smaller coherent topics, but rather causes an over-specialisation of them.

In summary, LDA still achieves the best scores overall, being often the first (or among the firsts) in ranking for every metric, whereas the other algorithms excel in particular contexts and can be specifically suitable for a given dataset. Increasing the number of topics is particularly helpful on bigger datasets, as it allows the topic models to find smaller yet more coherent subtopics within the collection, avoiding the drawback effect of being too specific. About label balance as tested through the Yahoo dataset, it appears that the balancing in the dataset has not a large impact in final results. On the contrary, training on the unbalanced version is often producing better coherence and V-measure. The reason can be found in the complete dropping of smaller categories, thus reducing the number of classes and achieving a higher-scoring topic/label mapping.

5.2 Varying the number of topics

To evaluate the effect of the choice of the number of topics (usually unknown beforehand), we train our models – except HDP, which infers the number of topics automatically – on 20NG using the same hyperparameters and varying only the number of topics. The results are shown in Table 2.

While there is a slight yet consistent improvement in the NPMI score for PVTM, we observe that increasing the number of topics does not consistently improve or hurt the coherence of the produced models. The fact that the score for 20 topics is usually the highest is probably due to the model finetuning, applied on this configuration. Finetuning every model for every number of topics requires a study of the co-optimisation of hyperparameters, which is out of the scope of this paper.

5.3 Varying the seed

For the models which allows to configure the random seed, we perform the evaluation on 20NG using the same hyperparameters except the seed (which we varied to have the values from 1 to 5). Even among 5 runs, we observe quite some variance in the metrics that is purely due to randomness which can be quite substantial. We report these results in Figure 2. While the effect is not very pronounced, it can be misleading. We thus recommend for topic models relying on random initialization to evaluate their models using different seeds, to guarantee a statistically significant comparison.

6 Conclusions and Future Work

In this work, we empirically evaluated 9 topic modelling algorithms using different coherence and ground-truth-based metrics on 3 text corpora reflecting a variety of properties, using a common evaluation framework. The results reveal several differences between the trained models, which obtain better or worse performances depending on the evaluation setting. Among these, LDA proves to be the most consistent performer overall, while embedding-based models prove to be less prone to generating meaningless topics.

The task of evaluating topic models remains a challenging one because of the inherent lack of a
ground-truth, the subjectivity of what constitutes a “coherent topic”, and the variety of settings wherein it is used. While every newly proposed topic model claims to improve on the existing state-of-the-art under some specific conditions, it is a worthwhile effort to revisit those claims and review them on a broader set of challenges and a unified pipeline, revealing their strengths and shortcomings. We also hope that by showing that no single metric can reflect the overall performance of any given topic model, we join a growing number of words drawing attention to the brittleness of most automatic metrics for topic models and the need of re-evaluating the standard practices of evaluation in the topic modelling literature.

As an extension to this work, we intend to study how other factors such as language, preprocessing and dataset characteristics can influence the performance on the metrics, as well as develop a unified protocol for evaluation that can allow us to draw more interesting insights into how the different topic modelling approaches fare in real use cases and downstream applications.

Acknowledgement This work has been partially supported by the French National Research Agency (ANR) within the ASRAEL project (grant number ANR-15-CE23-0018), by the European Union’s Horizon 2020 research and innovation program within the Odeuropa project (grant agreement No. 101004469), and by CHIST-ERA within the CIM-PLE project (CHIST-ERA-19-XAI-003).

References


Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A Novel Neural Topic Model and Its Supervised Extension. In AAAI Conference on Artificial Intelligence.


Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the Space of Topic Coherence Measures. In 8th ACM International Conference on Web Search and Data Mining (WSDM), page 399–408, New York, USA. ACM.


Jianhua Yin and Jianyong Wang. 2014. A Dirichlet Multinomial Mixture Model-Based Approach for Short Text Clustering. In 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), page 233–242, New York, USA. ACM.

Abstract
This article describes research on claim verification carried out using a multiple GAN-based model. The proposed model consists of three pairs of generators and discriminators. The generator and discriminator pairs are responsible for generating synthetic data for supported and refuted claims and claim labels. A theoretical discussion about the proposed model is provided to validate the equilibrium state of the model. The proposed model is applied to the FEVER dataset, and a pre-trained language model is used for the input text data. The synthetically generated data helps to gain information that improves classification performance over state of the art baselines. The respective F1 scores after applying the proposed method on FEVER 1.0 and FEVER 2.0 datasets are 0.65 ± 0.018 and 0.65 ± 0.051.

1 Introduction
Misleading claims and news are becoming pervasive in our lives. Sometimes these are extremely difficult to identify. As a result, they can cause serious problems. This makes the research on claim verification essential. Fake news can be broadly classified into three categories (Rubin et al., 2015): i) Serious fabrications (uncovered in mainstream or participant media, yellow press or tabloids); ii) Large-scale hoaxes; and iii) Humorous fakes (news satire, parody, game shows). To solve this problem, research on this subject has evolved from knowledge-base oriented methods to sophisticated deep learning-based techniques.

Related Work
In (Mihalcea and Strapparava, 2009), the authors used natural language processing (NLP) techniques to detect fake news. They used tokenization and stemming for preprocessing the data and applied Naive Bayes and Support Vector Machine (SVM) algorithms for classification. In recent research, the linguistic style and text source are considered the most critical factors to decide the genuineness of a fact or claim (Rashkin et al., 2017), (Baly et al., 2018), (Pérez-Rosas et al., 2017).

Sometimes multiple sources of particular claims are used as external resources for claim verification. In (Rashkin et al., 2017), researchers compared the linguistic characteristics of real news with satire, hoaxes, and propaganda. They presented a case study based on the data collected by PolitiFact.com, where they used Glove for embedding, and Long Short Term Memory (LSTM) for prediction. To improve their result, they concatenate the Linguistic Inquiry and Word Count (LIWC) features (Pennebaker et al., 2001) with LSTM output vectors before the activation layer.

LIWC features have played a vital role in claim verification research. LIWC extracts essential words that are part of psycho-linguistic categories and help in content analysis according to (Krippendorff, 2018; Neuendorf and Kumar, 2015). Their research work was extended by Kashyap et al. (Popat et al., 2018), who proposed an end-to-end framework for credibility analysis. This framework is capable of aggregating information from external evidence articles, the language of these articles, and the trustworthiness of their sources. It also generates informative features for user-comprehensible explanations (Popat et al., 2018).

Using external information sources is an effective technique for claim verification, e.g., researchers in (Pochampally et al., 2014), (Pasternack and Roth, 2011), (Ge et al., 2013), (Li et al., 2014), and (Wan et al., 2016) used external sources for similar types of tasks. Ravali et al. proposed a novel method based on correlations between different sources of news in (Pochampally et al., 2014). To find the correlation between sources, joint precision and joint recall are used.
Jeff Pasternack et al. introduced a generalized fact-finding framework in (Pasternack and Roth, 2011) to resolve conflicting claims. Similarly, (Ge et al., 2013), (Li et al., 2014), (Wan et al., 2016) also used potentially inconsistent sources and information to verify facts and claims. Liang Ge et al. (Ge et al., 2013) proposed a procedure that calculates the degree of information consistency, identifies the underlying reason(s) for any inconsistencies, and calculates a consistent score for each item. In (Li et al., 2014), researchers proposed an optimization framework in which truths and reliable sources are considered as two sets of unknown variables, and the framework aims to minimize the deviation between the truths and the multi-source observations. A generalized algorithm called TruthFinder is proposed in (Wan et al., 2016), which utilizes the information of different related websites to perform fact-checking.

In recent research on this topic, deep learning techniques are becoming popular. In (Choudhary and Arora, 2020), a sequential neural model is proposed, which helps to identify syntactic, grammatical, sentimental, and readability features for fake news detection. Yang et al. (Yang et al., 2018) proposed text and Image information based Convolution Neural Network (TI-CNN), which uses both text and images as evidence for fact-checking. In this model, CNN is used for feature extraction from both text and images.

Recently, the FEVER dataset has gained a lot of traction (Thorne et al., 2018b), (Thorne and Vlachos, 2019), (Thorne et al., 2019). Hence, we use FEVER for claim verification. In earlier research with FEVER, most researchers followed a pipeline suggested by the baseline model (Thorne et al., 2018a), which consists of three sequential phases. The phases are: identifying relevant Wiki articles, extracting the appropriate supporting sentences, and determining the truthfulness of the claim. Earlier researchers implemented the Wiki article phase by Wikipedia API, token matching techniques and the AllenNLP framework (Gardner et al., 2017). For sentence selection, most earlier researchers have used TF-IDF, sequence matching neural network, and some ranking methods. The classification task is done using a TF-IDF approach in the base model, however later on neural network models, natural language inference models, and deep learning models were used.

Here, a GAN (Goodfellow et al., 2014) based method is proposed for claim verification. This model is inspired by two GAN based Positive Unlabeled (PU) learning models such as GenPU (Hou et al., 2017) and Yang et al. (Yang et al., 2020). Fig. 1 shows the proposed model. This model has three subunits, \( P, N \), and \( L \). Each subunit consists of a generator \( (G_x) \) and discriminator \( (D_z) \) pair. Subunit \( P \) and \( N \) are responsible for generating positive and negative synthetic data; subunit \( L \) is responsible for binary class label generation of the synthetically generated data. Subunit \( P \) and \( N \) have positive \((X_p)\) and negative \((X_n)\) input data. The positive data consists of supported claims and respective evidence, while the negative data consists of refuted claims and respective evidence.

This model uses three generators \( (G_p, G_n, G_y) \) and three discriminators \( (D_p, D_n, D_y) \). \( G_p \) is responsible for generating positive claims and \( D_p \) discriminates between original and synthetically generated positive claims. \( G_n \) and \( D_n \) are responsible for similar functions for negative claims. \( G_y \) and \( D_y \) get the data generated by \( G_y \) and \( G_n \) and generate a class label \((0/1)\) and \( D_y \) is the discriminator for \( G_y \).

2 Proposed Methodology

As described above, three GAN units are used. These units are responsible for generating positive samples Equation 1, negative samples Equation 2 and class labels Equation 3. Algorithm 1 details the training of the generators and discriminators.

\[
\min_{G_p} \max_{D_p} V(D,G) = \mathbb{E}_{x \sim p_p(x)} \log(D_p(x)) + \mathbb{E}_{z \sim p_z(z)} \log(1 - D_p(G_p(z)))
\]
The proposed model can handle only supported and refuted claims. \( D_y \) is trained with both supported and refuted claims, while \( D_p \) and \( D_n \) are trained with only supported and refuted claims separately. Hence, \( D_y \) is a more powerful discriminator compared to \( D_p \) and \( D_n \). There is a possibility that \( D_p \) or \( D_n \) will assign some sentences generated by \( G_p \) and \( G_n \) wrongly. As \( D_y \) has the global view of both supported and refuted claims, it is better able to classify them. Consider a situation: \( G_p \) generates \( Y_p \) (a synthetic positive claim). In the next step, \( Y_p \) is the input to \( G_y \) and \( G_p \) is generating 1 (positive class label). The output of \( G_y \) and input of \( G_p \) is the input to the discriminator state \((D_y)\). If \( D_y \) classifies \( Y_p \) as real, then no penalty is incurred by \( G_y \) and \( G_p \) otherwise both \( G_p \) and \( G_y \) are penalized. Consider another situation, where \( G_y \) generates 0 (negative class label) for an input of \( Y_p \) and \( D_y \) also classifies the \( Y_p \) as fake, then a penalty will be added to \( G_p \), not \( G_y \). So \( D_y \) is acting as a global discriminator. Equation 4 is the loss function for the generator \( G_y \), where \( \pi_p \) and \( \pi_n \) are the probabilities of positive and negative claims in the dataset.

\[
L(y) = \pi_p[D_y(G_y(pz)) \log(D_y(G_y(pz)))) + (1 - D_y(G_y(pz)))] (1 - D_y(G_y(pz))) + \pi_n[D_y(G_n(z)) \log(D_y(G_n(z)))) + (1 - D_y(G_n(z)))]
\]

For a GAN, achieving equilibrium is very important. In the present context, to find the equilibrium condition, first, we need to find the optimal conditions for discriminators. Using the optimal conditions of the discriminators, the minimization conditions for the generator can be obtained. Considering the generators \((G_p, G_n, G_y)\) are fixed, and \( \pi_p \) and \( \pi_n \) are the probabilities of positive and negative claims in the dataset, at the equilibrium condition the distribution of positive generated data \((p_{gp}(x))\) and negative generated data \((p_{gn}(x))\) will follow the Equations 5 and 6, where \( p_{gp}(x) \) and \( p_{gn}(x) \) are the positive and negative class probability distributions.

\[
p_{gp}(x) = \pi_p(x)
\]

\[
p_{gn}(x) = \pi_n(x)
\]

The optimal discriminator functions \( D^*_p(x) \), \( D^*_n(x) \), \( D^*_y(x) \) can be derived by differentiating Equations 1, 2 and 3 (Hatua et al., 2021a).

\[
D^*_p(x) = \frac{p_p(x)}{p_p(x) + p_gp(x)}
\]

\[
D^*_n(x) = \frac{p_n(x)}{p_n(x) + p_{gn}(x)}
\]
Using Jensen–Shannon divergence (JSD) (Fuglede and Topsoe, 2004), we can show that the argmin generators are achieved when the following conditions are satisfied:

$$p_p(x) = p_{gp}(x) \quad (10)$$

$$p_n(x) = p_{gn}(x) \quad (11)$$

$$p_y(x) = \pi_p p_{gp}(x) + \pi_n p_{gn}(x) \quad (12)$$

### 3 Data

FEVER is a publicly available dataset for claim verification with three types of claims: i) supported, ii) refuted, iii) Not Enough Information (NEI). For every supported and refuted claim, there is supporting/refuting evidence, while for the NEI class there is no evidence. All evidence provided in the FEVER dataset is collected from Wikipedia. In most cases, the first few lines of a particular Wikipedia page are taken in FEVER dataset as the evidence. Table 1 shows two examples of claim, evidence pairs and their class labels. For the experiments, we used only Supported and Refuted claims.

FEVER training subset has 80,035 Supported claims, 29,775 Refuted claims, and 35,639 NEI claims. The FEVER 1.0 validation set and test set have 3,333 Supported claims, 3,333 Refuted claims, and 3,333 NEI claims respectively. FEVER 2.0 has 391 Supported claims, 396 Refuted claims, and 387 NEI claims respectively. For the experiments, we used only Supported and Refuted claims.

### 4 Experiments

The workflow of the experiment is given in Fig 2. In the first phase, data is preprocessed as described in Section 4.1. This preprocessed data is used as input to the proposed model for training. The Supported claim, evidence pairs are input to the positive synthetic data generator subunit, and the Refuted claim, evidence pairs are input to the negative synthetic data generator subunit. Once the proposed model is trained with the preprocessed data, the model is used for the testing phase using the test dataset. Finally, the model’s performance is compared with the results of other standard methods and SOTA models. The steps of the experiments are detailed below.

#### 4.1 Data preprocessing

For this experiment, only ‘Supported’ and ‘Refuted’ claims are considered from the training dataset. In the training dataset, every claim has one or more statements (evidence). For a particular claim, its corresponding statements are concatenated separately. For example, suppose claim (C) evidence (E) and label (L) are: \([C; E : < e_1, e_2, e_3 >, L]\). The input data format for subsequent processes will be: \(x = [< C; e_1, L >, < C; e_2, L >, < C; e_3, L >]\).

#### 4.2 GAN Implementation

The implementation of GAN is the central part of this research. Two types of GANs are implemented: text generating GAN and binary class label generating GAN. The text generating GANs generate synthetic text data for supported and refuted claims. The binary class label generating GAN generates the binary class label for each generated claim. To implement text generating GAN, we use LaTextGAN (Donahue and Rumshisky, 2018). LaTextGAN follows two phases for the implementation. During the first phase, it creates an encoded space, and in the second phase, it follows the traditional GAN (Goodfellow et al., 2014) implementation steps and generates synthetic data in the encoded space. Finally, the synthetically generated data is decoded into normal text data. On the other hand, the implementation of binary labels generating GAN is similar to the implementation of the traditional GAN (Goodfellow et al., 2014). The evidence for the synthetically generated sentences are selected from the Wikipedia database (Thorne et al., 2018b) using cosine similarity (Huang et al., 2008).
Claim: Tetris has sold millions of physical copies.
Evidence: It was announced that Tetris has sold more than 170 million copies, approximately 70 physical copies and ...
Label: True

Evidence: Roddick was ranked in the top 10 for nine consecutive years between 2002 and 2010, and won five Masters Series in that period.
Label: False

Table 1: Two claim, evidence pairs from FEVER

<table>
<thead>
<tr>
<th>Claim</th>
<th>Evidence</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tetris has sold millions of physical copies.</td>
<td>It was announced that Tetris has sold more than 170 million copies, approximately 70 physical copies and ...</td>
<td>True</td>
</tr>
<tr>
<td>Andy Roddick lost 5 Master Series between 2002 and 2010.</td>
<td>Roddick was ranked in the top 10 for nine consecutive years between 2002 and 2010, and won five Masters Series in that period.</td>
<td>False</td>
</tr>
</tbody>
</table>

In this case we have selected one evidence for every synthetically generated sentence. The synthetically generated data and the evidence are concatenated and processes following the steps mentioned in Section 4.1.

4.3 New GenPU Based Baselines

These baselines are inspired from the GenPU. To explore further we have modified GenPU in two variants: Inverted GenPU and Symmetric GenPU. In case of Inverted GenPU the value functions for the positive and negative text generating GAN are exchanged. Hence the respective value functions become the equations mentioned in Equation 13, 14 and 15.

\[
D^*_n = \arg\max_{D_n} \mathbb{E}_{x \sim p_p(x)} \log(D_n(x)) + \mathbb{E}_{z \sim p_z(z)} \log(1 - D_n(G_n(z)))
\]

\[
\min_{G_n} \max_{D_n} V(D, G) = - \mathbb{E}_{x \sim p_p(x)} \log(D_n(x)) - \mathbb{E}_{z \sim p_z(z)} \log(1 - D_n(G_n(z)))
\] (13)

\[
\min_{G_p} \max_{D_p} V(D, G) = \mathbb{E}_{x \sim p_p(x)} \log(D_p(x)) + \mathbb{E}_{z \sim p_z(z)} \log(1 - D_p(G_p(z)))
\]

(14)

\[
\min_{G_n} \max_{D_n} V(D, G) = \mathbb{E}_{x \sim p_p(x)} \log(D_n(x)) + \mathbb{E}_{z \sim p_z(z)} \log(1 - D_n(G_n(z)))
\]

(15)

In Symmetric GenPU the equations for both the value functions are same. The value functions for Symmetric GenPU are presented in Equation 16 and 17.

\[
\min_{G_p} \max_{D_p} V(D, G) = \mathbb{E}_{x \sim p_p(x)} \log(D_p(x)) + \mathbb{E}_{z \sim p_z(z)} \log(1 - D_p(G_p(z)))
\]

\[
\min_{G_n} \max_{D_n} V(D, G) = \mathbb{E}_{x \sim p_p(x)} \log(D_n(x)) + \mathbb{E}_{z \sim p_z(z)} \log(1 - D_n(G_n(z)))
\]

(16)

4.4 Other methods

The performance of the proposed method and new baselines is compared with other GAN based methods and classifiers. The GAN (LeakGAN (Guo et al., 2017) and LaTextGAN (Donahue and Rumshisky, 2018)) based models generate synthetic data and the synthetically generated data is added to the original dataset and it helps to create an extended feature space of the FEVER dataset and gives leverage to new features. This synthetically generated data is further classified using positive-unlabeled (PU) learning which considers supported facts as positive class and are added to the existing training dataset. Finally, this extended dataset is used for the training process. The synthetic data is generated using LeakGAN and LaTextGAN separately and two different sets of results are collected to compare the performance. The result of this method (Hatua et al., 2021b) for both the datasets is compared with the proposed method in Table 2, and Table 3. Other baselines include deep learning and machine learning based classification methods such as: BERT based classifier (Devlin et al., 2018), Graph Convolution Network (GCN) (Scarselli et al., 2008), Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), Convolution Neural Network (CNN) (Lawrence et al., 1997), Support Vector Machine (SVM) (Drucker et al., 1996), Naive Bayes (Lewis, 1998), Random forest (Pal, 2005), and Stochastic Gradient Descent (SGD) (Friedman, 2002).

To implement BERT based classifier Hugging-
face BERT (Devlin et al., 2018) pretrained transformer is used as tokenizer for the training, validation and testing dataset. The vocabulary size of the pretrained model is 30522 and the size of the hidden layer is 768. Later the pre-tuned model is fine tuned to classify the claims. In GCN, point wise mutual information between words is calculated to generate the graph. To implement the CNN five kernels of sizes 2, 3, 4, 5 and 6 are used. For LSTM the input data is encoded using GloVe (Pennington et al., 2014). The learning rate and batch size for GCN, CCN and LSTM are 0.001, 64 respectively. The Random forest is equipped with 1000 trees and the Scikit learn library (Pedregosa et al., 2011) is used for machine learning models.

5 Results

All models are trained with the FEVER training dataset and tested with FEVER 1.0 and FEVER 2.0 test dataset. In Tables 2, and 3 detailed results for each of the models are presented. Each experiment is repeated five times. The result for FEVER 1.0 is also compared with previous research work by Yang et al. (Yang et al., 2020).

<table>
<thead>
<tr>
<th>Classifiers</th>
<th>FEVER 1.0 Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>Precision: 0.45 ± 0.011, Recall: 0.44 ± 0.010, F1 Score: 0.44 ± 0.009</td>
</tr>
<tr>
<td>Leak GAN</td>
<td>Precision: 0.65 ± 0.003, Recall: 0.64 ± 0.006, F1 Score: 0.64 ± 0.003</td>
</tr>
<tr>
<td>LaTextGAN</td>
<td>Precision: 0.41 ± 0.008, Recall: 0.36 ± 0.016, F1 Score: 0.38 ± 0.009</td>
</tr>
<tr>
<td>GCN</td>
<td>Precision: 0.45 ± 0.015, Recall: 0.44 ± 0.013, F1 Score: 0.44 ± 0.013</td>
</tr>
<tr>
<td>SVM</td>
<td>Precision: 0.53 ± 0.013, Recall: 0.42 ± 0.013, F1 Score: 0.46 ± 0.013</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>Precision: 0.41 ± 0.016, Recall: 0.34 ± 0.014, F1 Score: 0.37 ± 0.015</td>
</tr>
<tr>
<td>RF</td>
<td>Precision: 0.33 ± 0.011, Recall: 0.33 ± 0.010, F1 Score: 0.33 ± 0.011</td>
</tr>
<tr>
<td>SGD</td>
<td>Precision: 0.31 ± 0.023, Recall: 0.22 ± 0.022, F1 Score: 0.25 ± 0.023</td>
</tr>
<tr>
<td>LSTM</td>
<td>Precision: 0.45 ± 0.003, Recall: 0.42 ± 0.004, F1 Score: 0.43 ± 0.004</td>
</tr>
<tr>
<td>CNN</td>
<td>Precision: 0.46 ± 0.012, Recall: 0.44 ± 0.011, F1 Score: 0.44 ± 0.012</td>
</tr>
<tr>
<td>Inverted GenPU</td>
<td>Precision: 0.52 ± 0.013, Recall: 0.71 ± 0.023, F1 Score: 0.60 ± 0.018</td>
</tr>
<tr>
<td>Symmetric GenPU</td>
<td>Precision: 0.33 ± 0.015, Recall: 0.54 ± 0.02, F1 Score: 0.40 ± 0.016</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>Precision: 0.50 ± 0.016, Recall: 0.93 ± 0.018, F1 Score: 0.65 ± 0.018</td>
</tr>
<tr>
<td>Yang et al. result</td>
<td>Precision: 0.61, Recall: 0.58, F1 Score: 0.60</td>
</tr>
</tbody>
</table>

In Tables 2, and 3 we see that the F1 score for the proposed method is better than the new baselines and previous research.

The gradual change of precision, recall, and F1 score for the FEVER 1.0 and FEVER 2.0 is presented in Fig. 3, and Fig. 4. Moreover, to visualize the distribution of original and synthetic data, t-SNE plots of the positive and negative generated data are shown in Figures 5, and 6. The perplexity of the t-SNE plot is 30, and the learning rate is 120. It can be observed that the distribution of synthetically generated data has less overlap with the original data.

Table 3: Result of FEVER 2.0

<table>
<thead>
<tr>
<th>Classifiers</th>
<th>FEVER 2.0 Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>Precision: 0.46 ± 0.013, Recall: 0.44 ± 0.014, F1 Score: 0.44 ± 0.013</td>
</tr>
<tr>
<td>Leak GAN</td>
<td>Precision: 0.52 ± 0.023, Recall: 0.51 ± 0.019, F1 Score: 0.51 ± 0.021</td>
</tr>
<tr>
<td>LaTextGAN</td>
<td>Precision: 0.42 ± 0.02, Recall: 0.39 ± 0.019, F1 Score: 0.40 ± 0.019</td>
</tr>
<tr>
<td>GCN</td>
<td>Precision: 0.43 ± 0.023, Recall: 0.39 ± 0.013, F1 Score: 0.40 ± 0.016</td>
</tr>
<tr>
<td>SVM</td>
<td>Precision: 0.40 ± 0.019, Recall: 0.37 ± 0.022, F1 Score: 0.38 ± 0.019</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>Precision: 0.33 ± 0.030, Recall: 0.22 ± 0.023, F1 Score: 0.26 ± 0.025</td>
</tr>
<tr>
<td>Random forest</td>
<td>Precision: 0.33 ± 0.014, Recall: 0.26 ± 0.017, F1 Score: 0.29 ± 0.015</td>
</tr>
<tr>
<td>SGD</td>
<td>Precision: 0.30 ± 0.025, Recall: 0.22 ± 0.029, F1 Score: 0.25 ± 0.027</td>
</tr>
<tr>
<td>LSTM</td>
<td>Precision: 0.43 ± 0.028, Recall: 0.40 ± 0.039, F1 Score: 0.41 ± 0.032</td>
</tr>
<tr>
<td>CNN</td>
<td>Precision: 0.41 ± 0.021, Recall: 0.38 ± 0.011, F1 Score: 0.39 ± 0.018</td>
</tr>
<tr>
<td>Inverted GenPU</td>
<td>Precision: 0.58 ± 0.024, Recall: 0.71 ± 0.022, F1 Score: 0.63 ± 0.012</td>
</tr>
<tr>
<td>Symmetric GenPU</td>
<td>Precision: 0.41 ± 0.016, Recall: 0.55 ± 0.011, F1 Score: 0.46 ± 0.013</td>
</tr>
<tr>
<td>Proposed method</td>
<td>Precision: 0.49 ± 0.061, Recall: 0.97 ± 0.041, F1 Score: 0.65 ± 0.051</td>
</tr>
</tbody>
</table>

Figure 3: Precision, Recall and F1 Score for FEVER 1.0 Dataset

Figure 4: Precision, Recall and F1 Score for FEVER 2.0 Dataset
generated positive data is very similar to that of original positive text data, while the distribution of the negative synthetic data is similar to the original negative text data. The positive synthetic data is much more similar to the positive text data compared to the similarity between negative synthetic data and negative text data.

The proposed GAN based model starts with some random values and tries to generate synthetic data, which helps to achieve a better F1 score. In the training process, after every epoch, we have calculated the F1 score for both the test datasets and observed a gradual improvement of the F1 score.

Fig. 7a, 7b, and 7c depicting the positive loss, negative loss and label generating loss. We can see the three losses are decreasing over epochs gradually,
which also suggests that all the generator discriminator pairs are training to achieve the equilibrium state. To test the gradual progression of the synthetically generated data, we also measure the similarity scores between original (positive and negative) data and synthetic data (positive and negative) while training the model. It has been observed that for the generated data, the similarity score gradually improves over epochs, as shown in Fig. 8, and 9. To measure the similarity 20,000 synthetically generated data are randomly selected and Cosine similarity (Singhal et al., 2001), Manhattan distance (Sinwar and Kaushik, 2014), Euclidean distance (Aggarwal et al., 2001) are calculated.

6 Conclusion

We propose a multiple GAN-based model that employs the GAN’s synthetic data generation capability to solve claim verification problems. The model generates synthetic data for supported, refuted claims and their class labels using three separate generator discriminator pairs. The synthetic data eventually helps in the fact-checking task for FEVER 1.0 and FEVER 2.0 test datasets. The results have shown that the proposed model starts with random data generation, and as the training progresses, it generates synthetic data similar to the original data.

Different statistical and analytical similarity metrics confirm that the similarity between original data and synthetically generated data increases as the training progresses. This gradual improvement of data quality shows the effectiveness of the model. The proposed model produces an F1 score of 0.65 ± 0.018 and 0.65 ± 0.051 for FEVER 1.0 and FEVER 2.0, respectively.

Dataset quality is a subtle issue, e.g., see (Verma et al., 2019; Verma and Marchette, 2019). In the future, this model can be extended to a multi-class classifier, and a similar set of experiments can be carried out on other publicly available standard datasets to test this proposed model’s effectiveness across different datasets.

Acknowledgments

Research was supported in part by grants NSF 1838147, NSF 1433817, ARO W911NF-20-1-0254 and ONR N00014-19-S-F009. Verma is the founder of Everest Cyber Security and Analytics, Inc. The views and conclusions contained in this document are those of the authors and not of the sponsors. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding copyright notation(s) herein.

References

Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. 2001. On the surprising behavior


Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. 2014. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1187–1198.


Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch:


Mengting Wan, Xiangyu Chen, Lance M. Kaplan, Jiawei Han, Jing Gao, and Bo Zhao. 2016. From truth discovery to trustworthy opinion discovery: An uncertainty-aware quantitative modeling approach. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1885–1894.


Semi-Supervised and Unsupervised Sense Annotation via Translations

Bradley Hauer, Grzegorz Kondrak, Yixing Luan, Arnob Mallik, Lili Mou
Alberta Machine Intelligence Institute, Department of Computing Science
University of Alberta, Edmonton, Canada
{bmhauer,gkondrak,yixing1,amallik,lmou}@ualberta.ca

Abstract

Acquisition of multilingual training data continues to be a challenge in word sense disambiguation (WSD). To address this problem, unsupervised approaches have been proposed to automatically generate sense annotations for training supervised WSD systems. We present three new methods for creating sense-annotated corpora which leverage translations, parallel bitexts, lexical resources, as well as contextual and synset embeddings. Our semi-supervised method applies machine translation to transfer existing sense annotations to other languages. Our two unsupervised methods refine sense annotations produced by a knowledge-based WSD system via lexical translations in a parallel corpus. We obtain state-of-the-art results on standard WSD benchmarks.

1 Introduction

Word sense disambiguation, the task of identifying the meaning of a word in context, is one of the central problems in natural language understanding (Navigli, 2018). It is a well-studied benchmark for evaluating contextualized representations of words (Loureiro et al., 2021), and is better understood than tasks such as WiC (Pilehvar and Camacho-Collados, 2019). Modern WSD methods can be divided into supervised and knowledge-based approaches. The former depend on sense-annotated corpora such as SemCor (Miller et al., 1994), while the latter rely instead on semantic knowledge bases such as WordNet (Miller, 1995).

While supervised WSD systems typically outperform knowledge-based systems (Scarlini et al., 2020b), their utility is limited by the availability of sufficiently large sense-annotated corpora for training. This includes systems based on contextualized embeddings (Bevilacqua and Navigli, 2020). In particular, there is a severe lack of high-quality sense-annotated corpora for languages other than English. This limitation has motivated the development of methods aimed at automatically disambiguating a large number of word tokens in a given unannotated corpus, ideally covering a wide range of word and sense types, while minimizing noise (Pasini and Navigli, 2017; Scarlini et al., 2019; Barba et al., 2020). The automatically tagged corpus can then be used to train a supervised WSD system, satisfying the dependency on training data without the need for manual annotation.

Following recent theoretical work Hauer and Kondrak (2020) on establishing the semantic equivalence of mutual translations, we introduce three translation-based methods for generating sense-tagged corpora. All three methods make use of lexical knowledge bases, and semantic information obtained from word-level translations. Semi-supervised LABELPROP creates a synthetic parallel corpus (bitext) by applying machine translation to a monolingual manually-annotated corpus, and projecting annotations to the target language. Similarly, unsupervised LABELGEN applies a knowledge-based WSD system to the English side of a bitext, and projects the resulting sense annotations across bitexts onto other languages. Finally, unsupervised LABELSYNC produces sense-annotated corpora in two languages at once by independently applying a knowledge-based WSD system to each side of a raw bitext, and then refining the initial annotations based on the confidence scores and multilingual information.

Our experiments on standard WSD test sets demonstrate that the new methods achieve state-of-the-art results in both semi-supervised and unsupervised sense annotation. We train two different reference supervised WSD systems on the generated data, and apply the resulting models to multilingual WSD benchmarks. Our results compare favourably to models trained on data produced by
the previous state-of-the-art sense annotation methods. Indeed, some of the results obtained with our unsupervised methods rival those obtained by training on a manually sense-annotated corpus.

Our contributions are as follows: We present three novel, scalable methods that can generate annotated corpora for any language for which a suitable lexical knowledge base is available. We show that these methods achieve state-of-the-art results on multiple languages. We make our code and corpora available.¹

2 Related Work

The sense tagging systems that we consider in this work, including our three novel methods, can be divided into four types according to two criteria (Figure 1). The first criterion is whether the method involves supervision in the form of a sense-annotated corpus. The second criterion is whether the method operates as a traditional self-contained WSD system, or instead assigns sense tags to a subset of the words in a corpus which can then be used to train a supervised WSD system. In this section, we discuss the most relevant examples of each of the four resulting types.

**Supervised WSD** systems rely on sense annotations to train disambiguation models, which are evaluated on benchmark datasets. Examples include GlossBERT (Huang et al., 2019), EWISE (Kumar et al., 2019), and EWISE (Bevilacqua and Navigli, 2020). Because they require labelled training data in the target language, such systems are generally impractical for languages other than English, nor are they directly comparable to our proposed methods.

**Knowledge-Based WSD** systems remain important due to the limited coverage of existing annotated corpora, as well as their English bias. These include graph-based systems such as UKB (Agirre et al., 2014), UKB enhanced with SyntagNet (Maru et al., 2019), and EWISE (Bevilacqua and Navigli, 2020). Because they require labelled training data in the target language, such systems are generally impractical for languages other than English, nor are they directly comparable to our proposed methods.

**Semi-Supervised Corpus Tagging** systems depend on sense annotated corpora in one language to produce sense annotations in other languages. The current state-of-the-art method in this setting is MuLaN (Barba et al., 2020), which propagates sense annotations from SemCor and WordNet Gloss Corpus (WNG, Langone et al., 2004) to semantically similar contexts in Wikipedia corpora using contextual word representations from mBERT. Our LABELPROP method differs by leveraging machine translation to directly propagate sense annotations across word alignment links.

**Unsupervised Corpus Tagging** systems produce sense annotations “from scratch”. Train-O-Matic (Pasini and Navigli, 2017) annotates Wikipedia in multiple languages by applying the Personalized PageRank (PPR) algorithm to BabelNet. OneSeC (Scarlini et al., 2019) combines Wikipedia categories and BabelNet synset representations to produce WSD training data, and outperforms Train-O-Matic. However, both Train-O-Matic and OneSeC annotate nominal instances only, and hence are not applicable to all-words WSD. On the other hand, EuroSense (Delli Bovi et al., 2017) jointly disambiguates content words of all parts of speech in a parallel corpus using a knowledge-based WSD system. Our LABELSYNC and LABELGEN methods differ in that they explicitly leverage lexical translation information obtained from a bitext.

![Figure 1: Typology of relevant sense tagging systems. Our own systems are shown in double-lined boxes.](https://www.cs.ualberta.ca/~kondrak)

Other work on using translations for WSD: Resnik and Yarowsky (1999) propose to distinguish senses only if a “minimum subset” of languages translate them differently. Apidianaki (2009) demonstrates how senses can be induced

¹https://www.cs.ualberta.ca/~kondrak
by clustering lexical translations, and proposes an unsupervised WSD system based on such induced sense inventory and translation information. Lefever et al. (2011) frame WSD as translation selection, and propose a method based on multilingual feature vectors. Finally, Taghipour and Ng (2015) annotate English words with their Chinese translations using manually crafted sense-to-translation mappings. These methods are not comparable with our work as they do not link their sense annotations to the WordNet sense inventory, and therefore are not applicable to modern WSD datasets.

3 Semi-Supervised LABELPROP

In this section, we introduce LABELPROP, a novel label propagation approach for constructing multilingual sense-annotated corpora. The idea is to translate a sense-annotated corpus in order to propagate the sense tags across the translations. No sense-annotated data is required in the target language. The method is composed of three steps: translation identification, knowledge-base filtering, and nearest neighbor filtering (Figure 2).

3.1 Translation Identification

Given a sense-annotated source corpus, we first translate the corpus into the target language using pre-trained neural machine translation models. Each sentence containing at least one source sense-annotated word is translated independently. If the translation of an annotated source word can be identified through word alignment, we annotate the translation with the same BabelNet synset as the aligned source word. This procedure is based on the assumption that lexical translations in context are semantically equivalent, and therefore very likely to express the same concept (Hauer and Kondrak, 2020).

For alignment, we use BABALIGN (Luan et al., 2020), a high-precision alignment tool which leverages translation information from BabelNet to improve on a base alignment system. In particular, BABALIGN augments the input corpus with lexical translation pairs to bias the aligner towards aligning words which are mutual translations. It also corrects alignments to maximize the number of aligned words that share BabelNet synsets. This emphasis on recovering word-level translation information makes BABALIGN particularly well-suited to our method.

3.2 Knowledge-Based Filtering

The sense-projection procedure in the previous step may annotate a word with a BabelNet synset which does not actually contain that word. These invalid sense annotations may occur due to non-literal translation (i.e., the word and its translation do not express the same concept), errors in translation or alignment, or omissions in BabelNet. Since each sense of a word must correspond to a specific synset, such invalid annotations are discarded.

3.3 Nearest Neighbor Filtering

In order to further increase the precision, we apply a semi-supervised WSD method to each target translation that is sense annotated by the previous steps. For each word, we verify that the annotation propagated from the source-language corpus matches the annotation assigned by the WSD system; otherwise, we discard that sense annotation.

Our semi-supervised WSD method uses a one-nearest-neighbor approach with ARES multilingual synset embeddings (Scarlini et al., 2020). We first obtain contextual word representations of each sense-annotated target translation by taking the sum of the last four layers of multilingual BERT (Devlin et al., 2019). Since ARES embeddings have twice the size of the original mBERT embeddings, we concatenate each obtained word representation with itself. We then compute the cosine similarity between the mBERT representation of the word, and the ARES representation of each synset containing the word. The synset that maximizes the similarity is taken as the output of this WSD system. To reiterate, we retain only the sense annotations from the previous step that agree with this WSD system.

4 Unsupervised Symmetric LABELSYNC

The LABELPROP method, presented in Section 3, is able to leverage existing sense annotated corpora, such as SemCor, to create comparable sense annotated corpora in other languages. However, the availability of sense-annotated corpora in other do-

Figure 2: LABELPROP propagates senses from language L1 to language L2.
mains and languages is very limited. On the other hand, large bitexts are relatively easy to obtain for many language pairs and domains.

To further reduce the dependency of WSD systems on any pre-existing annotated data, we introduce LABELSYNC, a method which annotates both sides of a given bitext. This method retains the idea of using word alignment to validate sense annotations, while eschewing the need for a sense annotated corpus. It is composed of three steps: monolingual word sense disambiguation, multilingual post-processing, and translation-based filtering (Figure 3).

4.1 Monolingual WSD
Our goal is to enrich both sides of the input bitext with sense tags. Since LABELSYNC does not assume access to any sense-annotated corpus, we employ a language-independent knowledge-based WSD system: a variant of UKB enhanced with SyntagNet (Maru et al., 2019). After each side of the bitext is annotated independently, we have two sense annotated corpora, one in each of the languages represented in the bitext.

4.2 Multi-Lingual Post-Processing
Now that both sides of the bitext are annotated independently, we leverage the lexical translation information inherent in the bitext to increase the accuracy of the sense annotations. To improve the performance of our base WSD system, we employ the SOFTCONSTRAINT method of Luan et al. (2020). This method is applicable to any base WSD system which assigns a numerical score, such as a probability, to each sense of a disambiguated word. Most modern WSD systems, including UKB, satisfy this property.

The SOFTCONSTRAINT method depends on word-level translations of each annotated word, as well as translation information from BabelNet, which is based on the hypothesis that the translation of a word token provides semantic information about its sense (Hauer and Kondrak, 2020). In our case, translation information is readily available from the bitext. As with our LABELPROP method, we use BABALIGN (Luan et al., 2020) to word align the bitext. For each sense-annotated token, the aligned word or phrase is treated as its translation.

The SOFTCONSTRAINT method can also incorporate sense frequency information, to bias the annotations toward more probable senses. However, we exclude sense frequency information from this step, as it provided no discernible benefit in our development experiments.

4.3 Translation-Based Filtering
In the final step, we aim to further reduce the noise in our sense-annotated corpora by employing a BabelNet-based filtering method, similar to the one described in Section 3.2. As before, the key idea is to impose two constraints on our sense annotations: (1) a word should only be annotated with a synset that contains the word, and (2) aligned words should be annotated with the same synset. LABELPROP initially guarantees only the latter constraint, so it has to discard some annotations to ensure the former. In contrast, LABELSYNC initially guarantees the first constraint, as UKB can only annotate a word with a synset containing it. However, since each side of the corpus is annotated independently, the second constraint may not hold. The final step of LABELSYNC is aimed at resolving this problem by synchronizing the sense annotations across both sides of the bitext.

Unlike Delli Bovi et al. (2017), who leverage embeddings of concepts to filter questionable annotations, we adopt a binary alignment-based criteria using the assumption of semantic equivalence of lexical translations. We retain only those annotations that refer to the same multilingual synset as the sense annotations of their translations. We also retain annotations if the token cannot be aligned, or if its translation is not annotated.

5 Unsupervised Asymmetric LABELGEN
Unlike LABELSYNC, our second unsupervised method, LABELGEN, assumes that the source language is English, and treats the two sides of the bitext differently. The goal is to leverage available English resources to improve WSD performance on other languages, rather than to generate English sense annotations.
5.1 English WSD
Given a bitext, we first apply a knowledge-based WSD system to the English side only, as described in Section 4.1. The lexical information for other languages is retrieved from BabelNet multi-synsets which are aligned to WordNet 3.0 synsets. This automatic candidate retrieval process is noisy, because most BabelNet lexicalizations are automatically generated from various resources. In addition, while English WordNet contains the sense frequency estimates from the manually-annotated SemCor, such information is not readily available for other languages. Hence, WSD annotations are more accurate for English compared to other languages.

5.2 Label Propagation
Having automatically sense-tagged the English side of the bitext, we propagate the labels to the non-English side using the procedure described in Section 3.1. In effect, we are applying the first part of LABELPROP, treating the English side as a sense-tagged corpus, and the other side as its translation. At the end of this process, both sides of the bitext are sense-annotated.

5.3 Re-Ranking and Filtering
We further refine the sense annotations on the non-English side of the bitext. We first apply the SOFTCONSTRAINT method as described in Section 4.2, which re-ranks the possible senses for each annotated word using the assigned WSD scores. We then apply the filtering procedure from Section 3.2, which removes any sense annotations that do not exist in the BabelNet sense inventory.

6 Evaluation
Following prior work, we extrinsically evaluate our corpus construction approaches by providing the generated annotations as training data for supervised WSD systems (reference systems), which are then evaluated on standard multi-lingual WSD benchmarks. While our methods could also be applied to low-resource languages, the current lack of evaluation datasets precludes such experiments in this work.

6.1 Reference Supervised WSD Systems
We perform experiments with two reference supervised WSD systems: (1) IMS (Zhong and Ng, 2010) with the most-frequent-sense (MFS) backoff for English, and (2) mBERT, a transformer-based method, built on multilingual BERT (Devlin et al., 2019), as described by Barba et al. (2020). We use the default parameter settings and number of training epochs. We train each model on each set of automatically produced sense annotations.

Following prior work, we use the SemEval-2007 dataset (Raganato et al., 2017) as our validation set for the English experiments. Because of the lack of standard validation sets for non-English languages, we use random samples of 1000 sentences from our training corpora. The hyperparameters of each system are held constant throughout all experiments.

6.2 Test Data
We test the reference WSD models on standard multilingual benchmark datasets: SemEval-2013 task 12 (Navigli et al., 2013), which contains data for Italian, Spanish, French, and German, and SemEval-2015 task 13 (Moro and Navigli, 2015), which covers Italian and Spanish. The SemEval-2013 datasets contain only nominal instances, while the SemEval-2015 datasets cover nouns, verbs, adjectives, and adverbs. We use the latest version of the datasets, which are annotated with synsets from BabelNet version 4.0.

For the experiments on English (Section 6.4.3), we use the standardized benchmarks of Raganato et al. (2017).

---

Table 1: Statistics of the sense-annotated corpora produced by each of our methods.

<table>
<thead>
<tr>
<th></th>
<th>LABELPROP</th>
<th>LABELSYNC</th>
<th>LABELGEN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Annotated Tokens</td>
<td>Annotated Word Types</td>
<td>Failed Alignments</td>
</tr>
<tr>
<td>EN</td>
<td>1,783,334</td>
<td>9,509</td>
<td>16,748</td>
</tr>
<tr>
<td>IT</td>
<td>399,569</td>
<td>25,361</td>
<td>29,290</td>
</tr>
<tr>
<td>ES</td>
<td>403,797</td>
<td>25,874</td>
<td>31,420</td>
</tr>
<tr>
<td>FR</td>
<td>407,590</td>
<td>25,193</td>
<td>32,181</td>
</tr>
<tr>
<td>DE</td>
<td>309,926</td>
<td>23,786</td>
<td>23,433</td>
</tr>
</tbody>
</table>

---

2https://github.com/edobobo/transformers-wsd
3https://github.com/SapienzaNLP/mwsd-datasets
et al. (2017)\footnote{http://nlp.uniroma1.it/wsdeval}, which comprise all-words test sets from five shared tasks: Senseval2 (Edmonds and Cotton, 2001), Senseval3 (Snyder and Palmer, 2004), SemEval-2007 (Pradhan et al., 2007), SemEval-2013 (Navigli et al., 2013), and SemEval-2015 (Moro and Navigli, 2015). We also report the average results on the concatenation of all five test sets, which we refer to as ALL.

### 6.3 Semi-Supervised Approaches

This section is devoted to the empirical evaluation of the LABELPROP method from Section 3.

#### 6.3.1 Experimental Setup

We apply LABELPROP to a sense-annotated English corpus comprised of SemCor (Miller et al., 1994) and the WordNet Gloss Corpus (WNG) (Langeone et al., 2004). Following Luan et al. (2020), we translate each sentence of our English corpus with Google Translate independently into Italian, Spanish, French, and German. As described in Section 3.1, we induce word alignment by applying BABALIGN with FASTALIGN (Dyer et al., 2013) as its base aligner. Table 1 contains the statistics for the corpora created with our methods. “Sense types” indicates the number of distinct word senses in the corpus. “Failed alignments” refers to the number of English sense annotations that could not be propagated.

We compare LABELPROP to MuLaN (Barba et al., 2020), the current state-of-the-art system for semi-supervised WSD. MuLaN uses SemCor+WNG as its manually-annotated base English corpus. Specifically, we apply the same procedure to train the supervised reference system (IMS or mBERT) on the annotated data produced by each method. We train a single model for each system, using only the corpus produced by that system for each language, which limits the impact of language-specific issues. Nevertheless, due to both software and hardware variables and hyper-parameters, our MuLaN results differ from those reported in the original paper.

When reporting the results achieved with mBERT, we also include the results of two recent WSD systems: ARES, using the reported results from Scarlini et al. (2020b), and 0-Shot WSD, with results replicated using the code provided by Barba et al. (2020). Since they are not designed to create annotated training data for other WSD systems, they are not directly comparable to LABELPROP.

#### 6.3.2 Results

Table 2 presents the multilingual WSD results obtained by IMS on standard test sets. While IMS is no longer a state-of-the-art system, it is still commonly used as a benchmark for evaluating automatically generated corpora (Scarlini et al., 2019). The results demonstrate the relative quality of the generated corpora: LABELPROP is better than MuLaN on Italian and Spanish, as well as on average. The difference in performance is found to be statistically significant across all six datasets ($p < 0.05$ with McNemar’s test). Neither of the approaches outperforms the most common sense (MCS) baseline on German, which we discuss further in Section 6.3.3. The ablation results in the last two rows show that both the nearest neighbour WSD filter (NN) and the translation-based filter (KB) improve the quality of the annotations.

Table 3 presents the corresponding results using the more recent mBERT as the reference system. Our results are slightly better on average than those of 0-shot and ARES. However, only the MuLaN results are directly comparable to our LABELPROP results, as both systems produce training data for a supervised WSD reference system. LABELPROP matches or outperforms MuLaN on every dataset except German SemEval-2013, and achieves better results on average compared to the results we replicated. The difference in F-score between LABELPROP and our replicated MuLaN experiment is significant for the SemEval 2015 Italian dataset ($p < 0.05$ with McNemar’s test).

<table>
<thead>
<tr>
<th>Model</th>
<th>SemEval-2013</th>
<th>SemEval-2015</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IT</td>
<td>ES</td>
</tr>
<tr>
<td>MCS</td>
<td>44.2</td>
<td>37.1</td>
</tr>
<tr>
<td>MuLaN</td>
<td>65.6</td>
<td>65.6</td>
</tr>
<tr>
<td>LABELPROP</td>
<td>71.4</td>
<td>71.0</td>
</tr>
<tr>
<td>-NN</td>
<td>70.5</td>
<td>70.1</td>
</tr>
<tr>
<td>-NN-KB</td>
<td>66.7</td>
<td>68.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>SemEval-2013</th>
<th>SemEval-2015</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IT</td>
<td>ES</td>
</tr>
<tr>
<td>ARES</td>
<td>73.0</td>
<td>75.3</td>
</tr>
<tr>
<td>0-shotSC-WNG</td>
<td>78.3</td>
<td>77.6</td>
</tr>
<tr>
<td>MuLaN</td>
<td>76.8</td>
<td>78.4</td>
</tr>
<tr>
<td>LABELPROP</td>
<td>78.4</td>
<td>78.5</td>
</tr>
</tbody>
</table>

Table 2: WSD F-score obtained with IMS trained on the corpora generated by MuLaN and LABELPROP.

Table 3: WSD F-score obtained with mBERT trained on the corpora generated by MuLaN and LABELPROP. Results of two semi-supervised WSD systems are included for reference.
6.3.3 Error Analysis

Error analysis suggests two reasons for the relatively low results on the German data. First, English multi-word compounds often correspond to single words in German, which makes it difficult to properly propagate English sense annotations. For example, the two words in *giveaway program*, which is a translation of *Werbeprogramm*, are separately annotated with different senses. The second issue is the quality of the BabelNet translation coverage. We observe that among 69,402 BabelNet synsets, that correspond to word senses appearing in SemCor+WNG, only 40,490 synsets contain at least one German translation, compared to over 50,000 synsets in each of the other three languages.

6.4 Unsupervised Approaches

In this section, we evaluate our unsupervised methods, LABELSYNC and LABELGEN, against comparable systems.

6.4.1 Experimental Setup

We adopt UKB (Agirre et al., 2014) as the base knowledge-based WSD system used in the first step of both LABELSYNC and LABELGEN to perform the initial tagging of a bitext. (This base WSD system is not to be confused with the reference supervised WSD system that is only used for the purpose of corpus evaluation.) Following Maru et al. (2019), we use WordNet as a lexical knowledge base, enriching it with information from WNG, and syntagmatic information from SyntagNet. BabelNet is the source of multilingual lexicalization information. When applying UKB, the PPR$_{w2w}$ variant of the personalized PageRank algorithm is run separately for each word, while concentrating the initial probability mass in the senses of the context words rather than the focus word.

Both of our unsupervised methods operate on an unannotated bitext. To keep the corpus size manageable, we randomly sample 200k sentences with English, French, German, Italian, and Spanish translations from EuroSense (Delli Bovi et al., 2017) discarding its existing sense annotations. This produces four bitexts with English as one of the languages, which we align at the word level using BABALIGN. The SoftCONsTRAINT method employed by LABELSYNC to refine the initial sense annotations leverages the lexical translations. Table 1 presents the statistics of the produced corpora.

<table>
<thead>
<tr>
<th>Model</th>
<th>SE-2013</th>
<th>SE-2015</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TT</td>
<td>ES</td>
</tr>
<tr>
<td>MCS</td>
<td>44.2</td>
<td>37.1</td>
</tr>
<tr>
<td>UKB+SyntagNet</td>
<td>72.1</td>
<td>74.1</td>
</tr>
<tr>
<td>SENSEMBERT</td>
<td>69.8</td>
<td>73.4</td>
</tr>
<tr>
<td>OneSeC</td>
<td>63.5</td>
<td>61.6</td>
</tr>
<tr>
<td>LABELSYNC</td>
<td>75.7</td>
<td>78.2</td>
</tr>
<tr>
<td>LABELGEN</td>
<td>77.8</td>
<td>80.5</td>
</tr>
</tbody>
</table>

Table 4: WSD F-score obtained with mBERT trained on the corpora generated by LABELSYNC and LABELGEN.

The direct competitor of LABELSYNC and LABELGEN is OneSeC (Scarlini et al., 2019), an unsupervised system which produces sense-annotated data by leveraging the semantic information within Wikipedia categories. Since OneSeC can only tag nouns, any model trained on a corpus it produces will likewise only be able to disambiguate nouns. Therefore, we do not apply models trained on OneSeC to the SemEval-2015 datasets, which include verb, adjective, and adverb instances. For our multilingual experiments, we also compare to two knowledge-based WSD systems described in Section 2: UKB with SyntagNet (Maru et al., 2019), and SENSEMBERT (Scarlini et al., 2020a).

6.4.2 Multilingual Results

Table 4 presents the multilingual WSD results when using mBERT as the reference WSD system. With the consistent exception of German, the results of mBERT trained on the annotations produced by LABELSYNC are substantially better than those trained on the corpus generated by OneSeC, which is the previous state-of-the-art for unsupervised corpora tagging. Unlike OneSeC, our unsupervised methods can annotate tokens representing all parts of speech, and can therefore be applied to the SemEval 2015 datasets. LABELSYNC also outperforms both knowledge-based WSD systems, UKB+SyntagNet and SENSEMBERT, and the most common sense (MCS) baseline. LABELGEN further improves on LABELSYNC by 1.8% on average. This makes it our best performing system, which sets a new state-of-the-art on the SemEval-2013 Italian, Spanish, and French datasets.

6.4.3 English Results

In this section, we evaluate LABELSYNC on English WSD. We do not test LABELGEN on English, as it was specifically designed to tag non-English corpora. Furthermore, because OneSeC annotates only nominal instances, we conduct separate all-
7 Conclusion

We have introduced new methods to address the knowledge acquisition bottleneck in word sense disambiguation in both the semi-supervised and unsupervised settings. The methods leverage recent advances in machine translation, alignment, and contextual embeddings. Extrinsic experiments with a variety of WSD systems demonstrate that the quality of the corpora created by our methods is substantially higher compared to those produced by prior work. Our methods for automatic sense tagging can produce annotated corpora for many languages, and approach the quality of manual annotation in some cases. We make our corpora available for further research.

One advantage of our unsupervised methods is that they can be applied to annotate any bitext involving any languages. We posit that our results could be further improved by annotating corpora with broader domain coverage, or by matching the domain of the source corpus to the domain of the data to be disambiguated. We leave this as a direction for future work.

Acknowledgments

The authors of this paper are listed in alphabetical order. Yixing Luan and Arnob Mallik conducted the experiments with the semi-supervised and unsupervised methods, respectively. Bradley Hauer and Grzegorz Kondrak prepared the final version of the paper.

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Alberta Machine Intelligence Institute (Amii).
References


Personality Predictive Lexical Cues and their Correlations

Xiaoli He
Rutgers University
New Brunswick, NJ, USA
hxl.daybreak@gmail.com

Gerard de Melo
Hasso Plattner Institute / University of Potsdam
Potsdam, Germany
http://gerard.demelo.org

Abstract

In recent years, a number of studies have used linear models for personality prediction based on text. In this paper, we empirically analyze and compare the lexical signals captured in such models. We identify lexical cues for each dimension of the MBTI personality scheme in several different ways, considering different datasets, feature sets, and learning algorithms. We conduct a series of correlation analyses between the resulting MBTI data and explore their connection to other signals, such as for Big-5 traits, emotion, sentiment, age, and gender. The analysis shows intriguing correlation patterns between different personality dimensions and other traits, and also provides evidence for the robustness of the data.

1 Introduction

The notion of personality refers to an individual’s characteristic patterns of thinking, feeling, and behaving (Sherman et al., 2013). Studies have shown that personality influences an individual’s language usage (Schwartz et al., 2013b; Tucker, 1968; Hirsh and Peterson, 2009). Hence, language may reveal subtle cues about an individual’s personality. Since an individual’s personality is known to be fairly stable across long periods of time, the relation between personality and language usage is expected to be analyzable given sufficiently large amounts of textual data.

Motivation. Many people now routinely post information pertaining to their daily life, thoughts, emotions, and opinions on different online social media platforms. A number of studies have shown that such social media text may enable automated personality predictions, as reviewed in more detail in Section 2. Yet, automatic personality prediction is still a challenging problem, and there remain several unresolved issues.

First, due to privacy concerns and the high labeling cost, the number of publicly available labeled datasets is limited, and the sample size in such datasets is often rather small (especially when compared with the high dimensionality of n-gram features). Beyond this, some datasets only provide a small number of sentences per sample. These limitations make it difficult to know to what extent results in individual studies generalize across different datasets.

Second, it is non-trivial to compare results across different studies, as they adopt different feature representations and machine learning methods, and consider different personality models. Two particularly well-known personality models are Myers-Briggs Type Indicators (MBTI) and Big-5 traits, which we introduce in more detail in Section 2. In the field of personality psychology, studies have shown clear correlations between self-reported MBTI and Big-5. For instance, MBTI’s INTROVERSION–EXTRAVERSION correlates with Big-5 EXTRAVERSION, MBTI’s SENSING–INTUITION and JUDGING–PERCEIVING correlate with Big-5’s OPENNESS trait, and MBTI’s JUDGING–PERCEIVING also correlates with Big-5 CONSCIOUSNESS (Tobacyk et al., 2008).

This raises the question of whether signals from naturally occurring text exhibit similar connections, and how they relate to other psychological and demographic variables.

Goals and Contributions. The goal of this paper is thus to empirically compare personality cues at the lexical level across different datasets, personality models, and methods. In our study we focus primarily on lexical signals based on multiple MBTI datasets from heterogeneous sources, which we compare against lexical cues for Big-5 traits. Additionally, we explore connections to sentiment and emotion lexicons, as well to demographic cues.
2 Background and Related Work

Personality Models. Different models have defined different traits (sub-dimensions) of personality. Two prominent ones are the MBTI and Big-5 schemes. The Myers-Briggs Type Indicator model (MBTI) (Myers et al., 1985) consists of the following four dimensions:

1. INTROVERSION–EXTRAVERSION (I–E): where a person focuses their attention;
2. INTUITION–SENSING (N–S): the way a person tends to take in information;
3. THINKING–FEELING (T–F): how a person makes decisions;
4. JUDGING–PERCEIVING (J–P): how a person deals with the world.

Numerous concerns have been raised regarding the validity and reliability of MBTI. For instance, MBTI assumes binary categorical labels for the aforementioned four dimensions, denoted e.g. as ESTJ, although the majority of people appear to exhibit a combination of different traits along a dimension. Still, MBTI is perhaps the most widely known model, and frequently mentioned in online profiles. In contrast, the Big-5 model (Goldberg, 1990) considers continuous scores along the following five dimensions:

1. EXTRAVERSION (extroversion): describes how outgoing and social a person is;
2. AGREEABLENESS: reflects how warm, friendly, and tactful a person is;
3. OPENNESS: considers how open-minded and authority-challenging a person is;
4. CONSCIENTIOUSNESS: reflects how self-disciplined and organized a person is;
5. NEUROTICISM (emotionism): indicates a person’s ability to remain stable and balanced.

Personality Assessment. In psychological assessments, personality is typically measured by means of standardized questionnaires that evaluate particular aspects of personality. This form of personality measurement has generally been found to be fairly stable and consistent. However, a major disadvantage is that experts first need to carefully compile long lists of questions, and individuals then need to explicitly fill out the questionnaire.

This has motivated research into computational analyses of naturally occurring text with the aim of obtaining automated assessments that correlate with the professional ones. In this regard, recent studies have considered several different social media platforms and personality scales. Different models have been developed, from simple Logistic or Linear Regression ones (Arnoux et al., 2017), support vector machines (Biel et al., 2013; Kumar and Gavrilova, 2019), to more complex models such as stability selection (Plank and Hovy, 2015), Gaussian process models (Arnoux et al., 2017), and ensemble methods aggregating multiple classifiers or regressors (Kumar and Gavrilova, 2019).

Past studies have also considered different features, such as word unigrams, word n-grams (Plank and Hovy, 2015; Yarkoni, 2010; Biel et al., 2013), and word embeddings (Arnoux et al., 2017; Siddique et al., 2019). For the studies using n-grams as features, some apply TF-IDF weighting schemes (Siddique et al., 2019; Biel et al., 2013), while others use unweighted features (Plank and Hovy, 2015; Yarkoni, 2010; Kern et al., 2014).

While the above studies have mostly sought to improve the performance of personality prediction on a given dataset using a variety of different methods and features, our study focuses on assessing the contribution of individual words and n-grams as signals for personality prediction, and their relationship to other lexical cues. In previous work, a few studies have focused on broader associations between personality and aggregate word categories (Yarkoni, 2010), such as Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2001). However, this may mask the contribution of individual words in the context of an open vocabulary scenario.

Lexical Analyses. Lexicon-driven analyses have proven fruitful in areas such as sentiment analysis (Ding et al., 2008; Mohammad et al., 2013; Kiritchenko et al., 2014; Islam et al., 2020) and emotion analysis (Kulahcioglu and de Melo, 2018; Raji and de Melo, 2020; Raji and de Melo, 2021), especially when there is no labeled data, as well as in social science and digital humanities (Pennebaker et al., 2001). With this approach, a dictionary of words (or bag of words) is generated, with a positive or negative value assigned to each word, reflecting the predictive power or correlation strength between the word and the specific target label or variable. Sap et al. (2014) explored lexical cues for age and gender. In traditional personality research, psychologists have developed closed-book vocabularies by self-rating on personality trait adjectives or verbs (Ashton et al., 2004b,a).

In light of the above, exploring automatically
induced lexical cues for personality prediction is a promising endeavor, and the resulting lexical signals can also be compared with lexical cues for other variables.

3 Lexical Cue Induction

In order to determine which words and n-grams are most correlated with specific personality variables, we assume a supervised learning setup with labeled training data that allows us to train a separate linear model for each target variable and identify salient lexical cues along with their weights. We compare several different variants with different feature representations and learning algorithms.

3.1 Feature Representations

Data Preprocessing. Our study considers only the linguistic information for each sample, along with the personality type labels, ignoring multimodal signals and metadata. The text is tokenized and the following preprocessing steps are applied:
1. Lower-casing;
2. Removing English stop words, tokens consisting only of numbers, and tokens mentioning personality types;
3. Replacing URLs, hashtags, usernames with ‘@URL’, ‘@HASHTAG’, ‘@USER’.

Feature Extraction. We extract unigram, 1-2 gram (unigram + bigram), and 1-2-3 gram (unigram + bigram + trigram) feature sets for each of the datasets. Due to the combinatorial explosion of n-grams, we apply a minimum frequency threshold, dropping any n-grams appearing less than 1% in each dataset. We further exclude tokens that consist solely of numbers. For the 1-2 grams and 1-2-3 grams, we also exclude tokens with punctuation in the first character or in the middle, such as (‘!’, ‘I’), (‘today’, ‘!’, ‘I’), (‘!’, ‘today’, ‘!’).

N-gram Weighting. Weighting is often used to adjust the importance of individual features. Besides using n-grams directly, we also used three types of weightings for each n-gram:
1. Relative term frequency, \( f_{\text{rel}}(w, d) = \frac{\text{freq}(w, d)}{\text{freq}(s, d)} \), is defined as the relative term frequency (TF) of a word \( w \) within a document \( d \).
2. TF-logIDF is a common definition of Term Frequency-Inverse Document Frequency (TF-IDF) weighting that incorporates a logarithmic scaling of IDF to dampen the effect of the ratio. In general, TF-IDF representations downweight words that appear universally across many documents, as these are less likely to be sufficiently discriminative in personality prediction.
3. TF-IDF differs from the above form in that we omit the logarithmical scaling of IDF.

3.2 Learning Algorithms

Linear models have often been used to induce weighted lexicons. Sap et al. (2014) compared the formula of linear multivariate models \( y = (\sum_j w_j x_j) + w_0 \) (summing over features \( f \)) with the use of a weighted lexicon \( L \) with term weights \( w_L(t) \) that is applied to a document \( d \) with frequencies \( f(t, d) \) as \( \sum_{t \in L} w_L(t) f(t, d) \). They prove that if relative term frequency is used as the feature representation, many multivariate modeling techniques can be viewed as learning a weighted lexicon plus an intercept.

Hence, the weight of a word in a lexicon can be obtained based on the coefficients from linear multivariate models. We thus treat each personality dimension as a distinct and independent classification or regression problem. For each combination of feature and weighting, we investigate four types of learning algorithms.

Stability Selection. In stability selection (Meinshausen and Bühlmann, 2010), the training data is repeatedly resampled in a bootstrap operation, and a model is learned for each such iteration, and features selected more frequently are presumed to be more robust indicators. As the base model, we use randomized logistic regression for MBTI datasets, and Randomized Lasso for the Big-5 dataset. We run 100 resampling procedures, such that on each resampling, 75% of the samples are randomly chosen. After the step of stability selection, we apply logistic regression for MBTI (linear regression for Big-5) on the selected features (n-grams), and save their coefficients.

Penalized Ridge Classification/Regression. We further consider Ridge Regression, i.e., linear least squares regression with L1 regularization. For MBTI, we apply Ridge Classification, i.e., the target classification is mapped to \( \{-1, 1\} \) so as to cast the problem as a regression task. The L1 penalty encourages sparse features, which is well-suited for our goal of identifying salient lexical cues. We split each dataset into training set and test set randomly with a ratio of 3:1 (also using...
the same ratio for the following two approaches).

**Penalized Support Vector Classification with Linear Kernel.** Support vector machines are well-suited for high-dimensional vector representations. Considering the high dimensionality and sparsity of our feature space, we consider support vector classification/regression with a linear kernel and L1 penalty in a 10-fold cross-validation setup.

**Penalized Multi-Layer Perceptrons.** Lastly, to better account for non-linearly separable data, we consider a feed-forward neural network with a 100-dimensional hidden layer and RelU activation function, trained using Adam optimization with an initial learning rate of 0.001.

## 4 Lexical Cue Analysis

In the following, we empirically assess lexical cues induced using the aforementioned techniques.

### 4.1 Datasets

Our analysis is based on 8 MBTI datasets and one Big-5 dataset, all consisting of naturally occurring English language text annotated with personality traits. For the former, we illustrate the respective data distributions in Figure 1. In particular, kaggle refers to the Kaggle Personality Cafe MBTI dataset, which provides 8,600 samples collected from the discussion forums of the Personality Cafe website. The Twitter datasets twitter_100g, twitter_500g, twitter_2000g are obtained from Plank and Hovy (2015). Each such dataset contains 1,500 samples, but they differ in the number of tweets per sample (100, 500, or 2,000). The reddit dataset is taken from Gjurković and Šnajder (2018), and provides 9,149 rows of comments from different Reddit authors with more than 1,000 words each. Due to the computational burden of the feature computation for bi- and tri-grams, we additionally also consider splits into smaller subsets (reddit0, reddit1, reddit2), which are mainly used for analysis (see the next section for further details).

Figure 1 shows the distribution within each dimension in the different MBTI datasets. For each dimension, the first type is coded as 0, and the second type is coded as 1. For example, for I–E, INTROVERT is represented as 0, and extravert is represented as 1. Figure 1 shows that, overall, each dataset has more INTROVERT and THINKING individuals. It has been reported that INTROVERT individuals prefer online communication (Plank and Hovy, 2015; Goby, 2006), though this overrepresentation may also have other causes. Interestingly, there are some differences between users in the different datasets. The Reddit data has more INTUITIVE users, while the Twitter data has more JUDGING users. The Reddit data includes slightly more users with the THINKING trait than the Twitter dataset.

![Figure 1: MBTI distribution on each dataset, where the dimensions I–E, N–S, T–F, and J–P are each mapped from 0 to 1](image)

We only considered a single Big-5 dataset, based on YouTube video blogs (Biel et al., 2013). The texts are the manually created transcripts, and the Big-5 score is not self-reported but rather the impression score assigned by a separate group of subjects, unlike most other datasets in our study.

### 4.2 Prediction Quality of Different Models

The prediction accuracies obtained for each combination of feature, weighting scheme, and model are given in Table 1, for each dimension of MBTI. The three numbers in a given cell represent the results from the three different weighting schemes: relative frequency, TF-logIDF, TF-IDF. Note that, owing to scalability considerations for trigrams, the 1-2-3 gram feature set was only considered for stability selection.

The results suggest that 1) the accuracy is fairly similar across different weighting schemes; 2) the accuracies consistently increase from unigrams to 1-2 grams, but only modestly with 1-2-3 gram features. We additionally plot the results using 1-2 grams weighted by TF-logIDF for each method in Figure 2. Each sub-figure shows the results from one model. Within each sub-figure, different bars indicate the results for different personality dimensions. Figure 2 conveys the following two messages: First, it shows that for the three linear models with penalty, dimension N–S obtains the highest accuracy, I–E the second highest, while J–P
and T–F exhibit lower accuracies, which are close to the baseline. This is consistent with the previous literature in that word usage usually has reliable predictions along INTROVERT—EXTRAVERT and SENSATION—INTUITION scales (Plank and Hovy, 2015; Kumar and Gavrilova, 2019), while showing worse performance on JUDGING–PERCEIVING and THINKING–FEELING.

Second, we observe that the performance of ridge regression, SVMs, and MLP are fairly consistent on different datasets. They perform better with the Reddit datasets, in comparison with the last three Twitter datasets. This may be due to the fact that the Reddit datasets have more samples (3,000 for reddit0/1/2 and 9,000 for reddit), while the Twitter datasets only have 1,500 samples. Additionally, the Reddit datasets have more words for each record. The high level of performance on the Kaggle Personality Cafe dataset (with its 8,600 samples) also accords with this hypothesis.

Overall, through our experiments on different datasets, we find that applying linear models on n-gram features consistently obtain fairly reliable predictions on at least two dimensions of MBTI, namely I–E and N–S. We have also run correlation analyses between each of two lexicons for the same dimension, and the results shows good correlation across different datasets and models. With the consistent performance across different models, we can confidently proceed to procure more robust lexicons across different datasets and methods.

### 4.3 Selecting Top-Ranked Features for MBTI

For each MBTI dimension, we have ~249 n-gram coefficient sets based on different datasets, features, weightings, and models. We select a small set of top-ranked lexical cues for each such dimension:

1. First, within each such lexicon, we normalize the coefficients using z-scores (enabling us to better compare them across different models).
2. Then, we sort the n-grams with the absolute values of their z-scores, and choose the top 75% n-grams – such that we obtain a subset \( X_i \) for each original set of features.
3. For each n-gram in \( X_i \), we calculated the term frequency across all feature sets, as well as the average z-scores, and chose the n-grams that appear in at least 60% among all sets.
4. Eventually, only the n-grams retained after the last step as well as their average z-score serve as the final set of weighted lexical cues for the dimension under consideration.

With the above procedure and the two filtering steps, we select small sets of top-ranked 79, 27, 124, 85 n-grams for I–E, N–S, T–F, J–P. Note that N–S has much fewer words, so we adjusted the thresholds in steps 2 and 3 (grid search in the two dimensional space with a step of 0.01), eventually using (0.8, 0.58) to obtain 85 n-grams for N–S.

Table 2 shows the top individual words for each dimension. Interestingly, it reflects certain stereotypical characteristics of each personality type. For example, EXTRAVERT individuals have more positive words such as *lol*, *haha*, *surprise*, while IN-
Table 1: Model accuracies on classification distinguishing I–E, T–F, N–S, and J–P, across different datasets using different features and weightings.

<table>
<thead>
<tr>
<th>Model</th>
<th>1-gram</th>
<th>2-grams</th>
<th>1-2-3 grams</th>
<th>Ridge 1-gram</th>
<th>SVM 1-gram</th>
<th>MLP 1-gram</th>
<th>SVM 1-gram</th>
<th>MLP 1-gram</th>
<th>SVM 1-gram</th>
<th>MLP 1-gram</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>kaggle_mbt</td>
<td>rosetta_1/f2k</td>
<td>rosetta_1/f2k</td>
<td>Stab 1-gram</td>
<td>Stab 1-gram</td>
<td>Stab 1-gram</td>
<td>Stab 1-gram</td>
<td>Stab 1-gram</td>
<td>Stab 1-gram</td>
<td>Stab 1-gram</td>
</tr>
<tr>
<td>I-E</td>
<td>0.82</td>
<td>0.87</td>
<td>0.83</td>
<td>[0.80, 0.80, 0.81]</td>
<td>[0.82, 0.83, 0.83]</td>
<td>[0.80, 0.80, 0.81]</td>
<td>[0.80, 0.80, 0.81]</td>
<td>[0.80, 0.80, 0.81]</td>
<td>[0.80, 0.80, 0.81]</td>
<td>[0.80, 0.80, 0.81]</td>
</tr>
<tr>
<td></td>
<td>0.80</td>
<td>0.86</td>
<td>0.83</td>
<td>[0.78, 0.78, 0.78]</td>
<td>[0.77, 0.77, 0.77]</td>
<td>[0.78, 0.78, 0.78]</td>
<td>[0.78, 0.78, 0.78]</td>
<td>[0.78, 0.78, 0.78]</td>
<td>[0.78, 0.78, 0.78]</td>
<td>[0.78, 0.78, 0.78]</td>
</tr>
<tr>
<td>N-S</td>
<td>0.83</td>
<td>0.85</td>
<td>0.86</td>
<td>[0.83, 0.83, 0.83]</td>
<td>[0.83, 0.83, 0.83]</td>
<td>[0.83, 0.83, 0.83]</td>
<td>[0.83, 0.83, 0.83]</td>
<td>[0.83, 0.83, 0.83]</td>
<td>[0.83, 0.83, 0.83]</td>
<td>[0.83, 0.83, 0.83]</td>
</tr>
<tr>
<td>J-P</td>
<td>0.83</td>
<td>0.85</td>
<td>0.86</td>
<td>[0.83, 0.83, 0.83]</td>
<td>[0.83, 0.83, 0.83]</td>
<td>[0.83, 0.83, 0.83]</td>
<td>[0.83, 0.83, 0.83]</td>
<td>[0.83, 0.83, 0.83]</td>
<td>[0.83, 0.83, 0.83]</td>
<td>[0.83, 0.83, 0.83]</td>
</tr>
</tbody>
</table>

TROVERT uses words expressing uncertainty such as awkward, probably, introvert. SENSING individuals focus on physical reality, while INTUITIVE individuals are driven by thoughts. Accordingly, the top words for S are concrete, such as soccer, jeans, cards, while for N they are more abstract, such as writing, science, proof. For F–T, the FEELING type has more adjectives describing feelings, e.g., wonderful, incredible, admirable, beautiful, while the THINKING type has words such as suppose, tastes, fix. For J–P, the words also reflect common stereotypes: career, passion, management, husband shows JUDGING individuals are more plan, work, and family oriented. Finally, the PERCEIVING type appears to use more words expressing feelings, such as sigh, jealous, wtf.

5 Correlation Analyses

Given the aggregated weighted lexical cues induced for each MBTI dimension, we seek to assess correlations with extrinsic lexical data covering a series of different phenomena.

5.1 Comparing Personality Models

The first interesting and straightforward comparison is between MBTI and Big-5.

First, we applied the same experiments on the YouTube dataset by Biel et al. (2013), so as to induce similar Big-5 signals, denoted as YouTube-B2013. Then, we ran Pearson correlation analyses comparing the two. We also compared our MBTI data with a well-established YouTube lexicon from Schwartz et al. (2013b), denoted as YouTube-S2013. The correlation results are given in Table 3. The analysis shows significant correlations between I–E and four dimensions of Big-5. J–P shows strong correlations with Agreeableness, Conscientiousness, Extraversiveness, Openness. T–F has a strong correlation with Agreeableness. Compared to Tobacyk et al. (2008), we have found more correlations between the two scales. In the personality literature in psychology, strong correlations have been found between Big-5 and MBTI. Most of the correlations found here can find support in psychology. The values given in brackets denote results that accord with significant correlations found in the psychology literature (Furnham, 1996).

The correlation between MBTI lexicons and our induced Big-5 lexicon (YouTube-B2013) is found to be much weaker. This is because this Big-5 lexicon is only based on one dataset (Biel et al., 2013), and that dataset has only around 400 samples.
Table 2: Top words (unigrams) for each dimension in the MBTI lexicons

<table>
<thead>
<tr>
<th>Dimension</th>
<th>I</th>
<th>E</th>
<th>S</th>
<th>N</th>
<th>F</th>
<th>T</th>
<th>P</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age 23–29</td>
<td>0.46</td>
<td>-0.43*</td>
<td>-0.12</td>
<td>-0.21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Age 30+</td>
<td>0.06</td>
<td>-0.11</td>
<td>0.43</td>
<td>0.17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Correlation between our MBTI lexicons and two YouTube Big-5 lexicons (Furnham, 1996)

<table>
<thead>
<tr>
<th>Correlation Type</th>
<th>I-E</th>
<th>J-P</th>
<th>N-S</th>
<th>T-F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extraversion</td>
<td>YouTube-S2013: 0.71**</td>
<td>-0.58**</td>
<td>0.65</td>
<td>-0.06</td>
</tr>
<tr>
<td></td>
<td>YouTube-B2013: -0.14</td>
<td>-0.24</td>
<td>0.17</td>
<td>0.29*</td>
</tr>
<tr>
<td>Agreeableness</td>
<td>YouTube-S2013: 0.52*</td>
<td>-0.77**</td>
<td>0.19</td>
<td>[0.84]**</td>
</tr>
<tr>
<td></td>
<td>YouTube-B2013: 0.03</td>
<td>-0.38</td>
<td>0.24</td>
<td>0.23</td>
</tr>
<tr>
<td>Openness</td>
<td>YouTube-S2013: [0.71]**</td>
<td>-0.71**</td>
<td>[0.24]</td>
<td>0.27</td>
</tr>
<tr>
<td></td>
<td>YouTube-B2013: 0.08</td>
<td>-0.47**</td>
<td>0.28</td>
<td>-0.07</td>
</tr>
<tr>
<td>Conscientiousness</td>
<td>YouTube-S2013: -0.19</td>
<td>-0.59**</td>
<td>0.14</td>
<td>[0.13]</td>
</tr>
<tr>
<td></td>
<td>YouTube-B2013: -0.09</td>
<td>-0.19</td>
<td>-0.41*</td>
<td>-0.01</td>
</tr>
<tr>
<td>Emotionism</td>
<td>YouTube-S2013: 0.75**</td>
<td>-0.07</td>
<td>-0.15</td>
<td>0.23</td>
</tr>
<tr>
<td></td>
<td>YouTube-B2013: 0.11</td>
<td>0.09</td>
<td>0.18</td>
<td>-0.20</td>
</tr>
</tbody>
</table>

* : p < .05  ** : p < .01

Table 4: Correlation between MBTI lexicons and emotion and sentiment

<table>
<thead>
<tr>
<th>Emotion</th>
<th>I-E</th>
<th>J-P</th>
<th>N-S</th>
<th>T-F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anger</td>
<td>-0.17</td>
<td>0.26</td>
<td>-0.23</td>
<td>-0.26</td>
</tr>
<tr>
<td>Fear</td>
<td>-0.26</td>
<td>0.41*</td>
<td>-0.18</td>
<td>-0.08</td>
</tr>
<tr>
<td>Joy</td>
<td>0.20</td>
<td>-0.15</td>
<td>-0.07</td>
<td>0.28**</td>
</tr>
<tr>
<td>Sadness</td>
<td>-0.12</td>
<td>0.41*</td>
<td>0.10</td>
<td>-0.15</td>
</tr>
<tr>
<td>Arousal</td>
<td>0.22**</td>
<td>0.00</td>
<td>-0.10</td>
<td>0.01</td>
</tr>
<tr>
<td>Dominance</td>
<td>0.26**</td>
<td>-0.15**</td>
<td>-0.27**</td>
<td>0.08</td>
</tr>
<tr>
<td>Valence</td>
<td>0.15**</td>
<td>0.02</td>
<td>0.01</td>
<td>0.28**</td>
</tr>
<tr>
<td>Sentiment (Ding et al., 2008)</td>
<td>0.25**</td>
<td>-0.13*</td>
<td>-0.25**</td>
<td>0.25**</td>
</tr>
<tr>
<td>Sentiment (NRC)</td>
<td>0.16**</td>
<td>-0.21**</td>
<td>0.02</td>
<td>0.27**</td>
</tr>
<tr>
<td>Sentiment (Twitter)</td>
<td>0.16*</td>
<td>-0.24**</td>
<td>-0.10</td>
<td>0.43**</td>
</tr>
<tr>
<td>Sentiment (VADER)</td>
<td>0.36**</td>
<td>-0.41**</td>
<td>-0.17</td>
<td>0.42**</td>
</tr>
</tbody>
</table>

* : p < .05  ** : p < .01

Table 5: Correlation between MBTI signals with demographic signals

<table>
<thead>
<tr>
<th>Demographic</th>
<th>I-E</th>
<th>J-P</th>
<th>N-S</th>
<th>T-F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>-0.03</td>
<td>-0.12**</td>
<td>0.03</td>
<td>-0.05</td>
</tr>
<tr>
<td>Gender</td>
<td>-0.09*</td>
<td>-0.10*</td>
<td>0.13*</td>
<td>0.23**</td>
</tr>
<tr>
<td>Gender (1-2-3-grams)</td>
<td>0.09</td>
<td>-0.08**</td>
<td>0.72**</td>
<td>0.73**</td>
</tr>
</tbody>
</table>

* : p < .05  ** : p < .01

525
5.2 Correlation with Emotion and Sentiment

Personality influences an individual’s emotions, opinions, and behaviours. This motivates us to study the relationship between MBTI cues and other psychological lexicons, such as sentiment and emotion ones. Little work has been conducted on the correlation between personality and emotion, but the definitions of the MBTI dimensions suggest a possible connection.

We retrieved several emotion and sentiment lexicons and computed their correlation with our data in Table 4. The first four lexicons (Mohammad, 2017) focus on four basic affective categories from the Plutchik model (Plutchik, 1980): anger, fear, joy, and sadness. Only J–P has a positive correlation with fear and sadness, while T–F has a positive correlation with joy. This suggests that the FEELING type tends to use more joy-related words, while PERCEIVING individuals tend to use more fear and sadness related words. The next three lexicons (Mohammad, 2018) are based on the PAD model (Russell and Mehrabian, 1977), which conceptualizes emotion along three dimensional axes – arousal, dominance, and valence. We observe that I–E has significant positive correlations with all three dimensions, reflecting that EXTRAVERTs focus more on outside stimuli, and tend to have more emotional reactions. J–P and N–S are negatively correlated with dominance, meaning that the JUDGING and INTUITION types exhibit higher dominance – i.e., more stability. These two types perhaps also tend to analyze and give solutions, while SENSING and PERCEIVING individuals exhibit stronger feelings, which leads to lower dominance. T–F shows positive correlation with valence – suggesting that the FEELING type may have more emotional reactions.

As personality affects an individual’s way of writing and talking, we further hypothesize that people with the same personality may tend to use expressions with similar sentiment. Thus, we also compare our MBTI data with three sentiment lexicons: Ding et al. (2008), NRC (Mohammad et al., 2013), Twitter (Kiritchenko et al., 2014), and VADER (Hutto and Gilbert, 2014). Both I–E and T–F show positive correlations with sentiment, while J–P shows a negative correlation. This suggests that EXTRAVERT, FEELING, and JUDGING types may tend to have more positive sentiment. Lin et al. (2017) developed a Big-5 personality-based sentiment classifier and argue that it performs better than an ordinary sentiment classifier, providing further corroboration for a potential correlation between personality and sentiment analysis.

5.3 Correlation with Demographic Signals

Twitter and Reddit have large user bases, including different gender and age groups. We study potential correlations between these two demographic features and the MBTI dimensions, relying on the age and gender lexicons by Sap et al. (2014), as well as the age-specific gender lexicons by Schwartz et al. (2013a). The correlations are reported in Table 5. Only J–P has a negative correlation with the age lexicons. It appears plausible that older individuals might rely more on judgement than perception.

The two general gender lexicons gender and gender (1-2-3-grams) show that I–E has a slightly negative correlation with gender, J–P has a strong correlation, while N–S and T–F have moderate positive correlations with gender. Note that for the gender lexicons, male is here treated as negative, and female as positive, and the two lexicons fail to account for other gender identities. The correlation analysis is consistent with the stereotypes that female users tend to use more words about feelings (F) and are more sensible (S) in general. However, it is interesting to see that female users are found to be more JUDGING. When we control for age group, most correlations between gender and personality disappear, and only J–P showed strong positive correlation with gender in the age group 13 to 18, and negative correlation in age group 23 to 29.

6 Conclusion

We have inferred personality predictive lexical signals, i.e., words and n-grams along with their weights, for each MBTI dimension. The data is induced based on several diverse MBTI datasets, using a variety of feature sets, weighting schemes, and learning algorithms. Our focus here is on identifying correlations with other kinds of cues, including Big-5 data, as well as emotion, sentiment, and gender-predictive lexicons. We show that naturally occurring text harbors subtle cues exhibiting correlations that largely accord with findings from psychology on self-reported personality correlations. This provides further evidence for the validity of drawing on such naturally occurring data for automated lexical cue induction.
Ethical Statement

It is important to keep in mind that all results presented here are highly dependent on the characteristics of the respective datasets and on the lexicon induction methodology. As shown in Section 4.1, different datasets provide data from different sources, leading to biases both in the kinds of textual content they provide and in the label distributions. Additionally, using automated predictors for lexicon induction tends to lead to signals reflective of particularly stereotypical cues, and linear models are unable to account for the particular context of a particular word mention. Thus, the particular word-level correlations observed in this study do not entail that such correlations also hold among people exhibiting a particular trait. Last but not least, mere correlations such as those considered in this paper do not license conclusions about particular individuals or groups of individuals, and any studies attempting to predict the personality of individuals or groups of individuals would need to consider a large number of very serious ethical and privacy concerns.

References


Barbara Plank and Dirk Hovy. 2015. Personality traits on twitter—or—how to get 1,500 personality tests in a week. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 92–98.


Tal Yarkoni. 2010. Personality in 100,000 words: A large-scale analysis of personality and word use among bloggers. *Journal of research in personality*, 44(3):363–373.

variation in well-being using tweets. In *Proceedings of the 7th International AAAI Conference on Weblogs and Social Media*, ICWSM.
Evaluation Datasets for Cross-lingual Semantic Textual Similarity

Tomáš Hercig†
† Department of Computer Science and Engineering, University of West Bohemia, Plzeň, Czech Republic
tigi@ntis.zcu.cz

Pavel Král†‡
‡ NTIS – New Technologies for the Information Society, University of West Bohemia, Plzeň, Czech Republic
pkral@kiv.zcu.cz

Abstract

Semantic textual similarity (STS) systems estimate the degree of the meaning similarity between two sentences. Cross-lingual STS systems estimate the degree of the meaning similarity between two sentences, each in a different language.

State-of-the-art algorithms usually employ a strongly supervised, resource-rich approach difficult to use for poorly-resourced languages. However, any approach needs to have evaluation data to confirm the results. In order to simplify the evaluation process for poorly-resourced languages (in terms of STS evaluation datasets), we present new datasets for cross-lingual and monolingual STS for languages without this evaluation data. We also present the results of several state-of-the-art methods on these data which can be used as a baseline for further research.

We believe that this article will not only extend the current STS research to other languages, but will also encourage competition on this new evaluation data.

1 Introduction

Recently, research in natural language understanding is moving beyond monolingual solutions (Wada et al., 2019; Conneau and Lample, 2019; Lin et al., 2019). However, any solution needs some kind of evaluation data. The common source of evaluation data for semantic meaning comparison are the STS tasks from SemEval workshop.

The STS task has a long history at SemEval workshops. Since 2012 till 2017 the STS task has been held annually creating a considerable amount of evaluation datasets. However, most of these datasets are in English and Spanish.

This has led us to the creation of new cross-lingual evaluation data for STS because all algorithms need verification e.g. on evaluation data.

The evaluation data consist of pairs of sentences and the degree of their semantic similarity.

This paper presents new STS datasets and thorough experiments using linear transformations for cross-lingual STS. We see four main contributions of our work:

• We show an overview of the existing cross-lingual STS datasets.
• We present newly created datasets for both cross-lingual and monolingual STS.
• We extend experiments for previously published methods replicating the results and validating the conclusions.
• We present initial experiments and provide baseline results on the new datasets.

2 Related Work

This section presents related datasets for STS with main focus on cross-lingual datasets.

Even though most of the datasets is in English and Spanish, there are also available monolingual datasets in Arabic and Czech.

The cross-lingual datasets are also quite focused on English and usually one side of the sentence pair is in English. In summary, the cross-lingual datasets are available in English paired with Arabic, Croatian, Czech, Italian, Spanish, and Turkish. More detailed overview of these datasets can be found further in this section.

In Section 3, we present datasets for both monolingual and cross-lingual STS in Czech, English, French, and German.

• SemEval 2012: Agirre et al. (2012) introduced the shared task competition in English. They provided five datasets (paraphrase sentences MSRpar, video descriptions MSRvid,
automatically translated sentences MTnews and MTeuroparl, and gloss pairs OnWN) consisting of 2234 training sentence pairs, 3108 testing sentence pairs, and 6 trial pairs. Gluvaš et al. (2017) translated one side (750 sentences each) of two English monolingual datasets (MSRvid and OnWN) from SemEval 2012 task 6 to Spanish, Italian, and Croatian.

• SemEval 2013: Agirre et al. (2013) continued with English STS core task. The whole dataset contains 2250 test and 20 trial new sentence pairs from four datasets (news Headlines, mapping of lexical resources OnWN and FNWN, machine translation evaluations SMT). They also introduced typed similarity task for predefined types (e.g. author, location, subject and description).

• SemEval 2014: Agirre et al. (2014) divided the task into two subtasks one for English (3750 sentence pairs) and the other for Spanish. The English data contains image descriptions (Images), news headlines (Headlines), gloss pairs (OnWN), news title and tweet comments (Tweet-news), discussion forum and news (Deft-forum and Deft-news). The newly introduced Spanish dataset contained 480 sentence pairs from Wikipedia, 324 sentence pairs from Google News and 65 trial sentence pairs.

• SemEval 2015: Agirre et al. (2015) continued with both English (3000 test pairs, 70 trial pairs) and Spanish (751 pairs) subtasks and newly introduced an interpretable STS subtask. The interpretation is evaluated on the alignments of sentence pairs. The English dataset contains image descriptions (image), news headlines (headlines), student and reference answers (answers-students), answers from exchange forums (answers-forum), and discussion forum comments (belief).

• SemEval 2016: Agirre et al. (2016) proposed two Spanish-English datasets in SemEval 2016 task 1. One consists of news headlines News (301 sentence pairs) and the other contains sentences from multiple sources including news headlines, question-answering, plagiarism detection, etc. Multi-source (294 sentence pairs). Gluvaš et al. (2017) translated Spanish sentences from both datasets to Italian and Croatian.

• SemEval 2017: Cer et al. (2017) introduced four cross-lingual datasets and three monolingual datasets. Each dataset contains 250 sentence pairs and the data source is Flickr30k image captions (Young et al., 2014), except Track4b which is based on data from WMT 2014 quality estimation task (Bojar et al., 2014). The cross-lingual dataset contains the following language pairs: Spanish-English (Track4a and Track4b), Arabic-English (Track2), and Turkish-English (Track6). The monolingual dataset consists of English (Track5), Spanish (Track3), and Arabic (Track1).

• Czech STS: Svoboda and Brychcín (2018) translated the English Images and Headlines parts of the dataset from SemEval 2013 - 2015 resulting in a dataset of 575 pairs of news headlines and 850 pairs of image descriptions. However, the links to the English dataset were not preserved leaving us only with a Czech monolingual dataset.

3 Dataset

<table>
<thead>
<tr>
<th></th>
<th>First</th>
<th>Second</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CS</strong></td>
<td>1612</td>
<td>1576</td>
</tr>
<tr>
<td><strong>DE</strong></td>
<td>2002</td>
<td>2010</td>
</tr>
<tr>
<td><strong>EN</strong></td>
<td>2179</td>
<td>2186</td>
</tr>
<tr>
<td><strong>FR</strong></td>
<td>2382</td>
<td>2245</td>
</tr>
</tbody>
</table>

Table 1: Number of tokens for each dataset.

We used the English monolingual dataset (Track5) from SemEval 2017 task 1 (Cer et al., 2017) to create new evaluation data for Czech, French and German.

We translated (using Google Translate) and then manually checked both sides of 250 pairs from the English STS dataset from SemEval 2017 (Track5) into Czech, French, and German.

The translations were manually checked by two upper intermediate (B2) level speakers of the given language in case of French and German and native speakers of Czech. We also asked them to preserve the meaning of the translation as much as possible in relation to the semantic similarity score.

We assume that the translated pairs preserve the same semantic similarity score. The resulting
dataset thus contains monolingual and cross-lingual datasets for all language pairs.

Table 1 shows number of tokens in a dataset per side (part) of the evaluation pair.

The dataset consisting of 2000 sentences is available for research purposes at https://gitlab.com/tigi.cz/cross-lingual-sts.

Each file contains one sentence per line. The STS evaluation pair consists of the first and second part (sentence) each stored in a separate file. The semantic similarity score is located in the gold score file. The lines in all files correspond with each other. To get the evaluation data for e.g. EN-CS load the file STS.2017.input.track5.EN.first.txt and STS.2017.input.track5.CS.second.txt and the gold standard in file STS.2017.gs.track5.first-second.txt.

This new resource consists of four monolingual datasets and twelve cross-lingual datasets.

4 Experiments

We follow Brychcín (2020) who used bilingual dictionaries and a new transformation for word level semantic representations which reduces hubness in semantic spaces. He also evaluated his methods on several STS datasets including cross-lingual.

The transformations of semantic spaces and combinations of word representations are too complex and beyond the scope of this short paper, for a thorough description of these methods please refer to the original publication.

In the replicated experiments on new datasets we use monolingual semantic spaces transformed into a unified space using bilingual dictionaries. STS performance is measured by the Pearson correlation between automatically estimated scores and human judgments.

Our experiments start with building monolingual semantic spaces for each of tested languages, namely, Czech (CS), German (DE), English (EN), and French (FR). For all languages we use character-n-gram-based skip-gram model (Bojanowski et al., 2017) pre-trained on Wikipedia1.

For each language, we construct the vocabulary from 300k most frequent words. We estimate IDF weights on the Wikipedia corpus for every language. Each Wikipedia article represents a document.

The bilingual dictionaries between each pair of languages are created from the 20k most frequent words in the corpus using Google translate.

The global post-processing techniques for semantic spaces used by Brychcín (2020) consist of two steps column-wise mean centering and word vector normalization to unit vectors. This guarantees that all word pairs in the dictionary contribute equally to the optimization criteria of the linear transformation. We always apply this post-processing for both semantic spaces before the linear mapping.

4.1 Linear Transformations

We experiment with the following five techniques for linear mapping to transform the semantic spaces. For detailed description of these methods please see (Brychcín, 2020).

- Least Squares Transformation (LS)
- Orthogonal Transformation (OT)
- Canonical Correlation Analysis (CCA)
- Ranking Transformation (RT)
- Orthogonal Ranking Transformation (ORT)

4.2 Word Combinations

Semantic textual similarity is estimated by combining word representations by Linear Combination (LC), Principal Angles (PA)2, and Optimal Matching (OM)3. We evaluate both uniform weighting (for mutual comparison with original methods) and IDF weighting in all three STS approaches.

RT and ORT require special settings to work properly we use the same settings as Brychcín (2020).

4.3 Results

Table 3 shows the mean Pearson correlations for each linear transformation combined with different STS techniques on the created cross-lingual STS datasets. We can state our results support the claims by Brychcín (2020). ORT outperformed other transformations independently of STS technique. IDF weighting boosts the correlations in all cases and together with OM yields the best performance.

In Tables 2, and 4 we show correlations achieved by the best settings, i.e., OM with IDF weighting. In table 2 we compare our results with the top performing system ECNU (Tian et al., 2017) and

1Available at https://fasttext.cc.

2Using $r = 4$ as recommended by Mu et al. (2017)

3For detailed description see(Sultan et al., 2015; Glavaší et al., 2017; Brychcín, 2020)
Table 2: Individual Pearson correlations for monolingual datasets using OM with IDF weighting.

<table>
<thead>
<tr>
<th></th>
<th>CS-CS</th>
<th>DE-DE</th>
<th>EN-EN</th>
<th>Fr-Fr</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Results</td>
<td>0.736</td>
<td>0.706</td>
<td>0.786</td>
<td>0.702</td>
<td>0.732</td>
</tr>
<tr>
<td>Tian et al. (2017)</td>
<td>-</td>
<td>-</td>
<td>0.852</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Cer et al. (2017)</td>
<td>-</td>
<td>-</td>
<td>0.728</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3: The mean Pearson correlations over monolingual and cross-lingual datasets. The highest correlations are in bold.

<table>
<thead>
<tr>
<th></th>
<th>LC</th>
<th>LC IDF</th>
<th>PA</th>
<th>PA IDF</th>
<th>OM</th>
<th>OM IDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monolingual</td>
<td>0.544</td>
<td>0.659</td>
<td>0.695</td>
<td>0.715</td>
<td>0.691</td>
<td>0.732</td>
</tr>
<tr>
<td>Cross-lingual</td>
<td>LS</td>
<td>0.032</td>
<td>0.253</td>
<td>0.382</td>
<td>0.486</td>
<td>0.478</td>
</tr>
<tr>
<td></td>
<td>CCA</td>
<td>0.088</td>
<td>0.319</td>
<td>0.373</td>
<td>0.503</td>
<td>0.498</td>
</tr>
<tr>
<td></td>
<td>OT</td>
<td>0.140</td>
<td>0.361</td>
<td>0.416</td>
<td>0.524</td>
<td>0.511</td>
</tr>
<tr>
<td></td>
<td>RT</td>
<td>0.186</td>
<td>0.385</td>
<td>0.460</td>
<td>0.531</td>
<td>0.519</td>
</tr>
<tr>
<td></td>
<td>ORT</td>
<td>0.320</td>
<td>0.464</td>
<td>0.519</td>
<td>0.560</td>
<td>0.556</td>
</tr>
</tbody>
</table>

Table 4: The mean Pearson correlations over language pairs of cross-lingual datasets using OM with IDF weighting. The result for CS-DE is the mean value of CS-DE and DE-CS.

<table>
<thead>
<tr>
<th></th>
<th>CS-DE</th>
<th>CS-EN</th>
<th>CS-FR</th>
<th>DE-EN</th>
<th>DE-Fr</th>
<th>En-Fr</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>LS</td>
<td>0.544</td>
<td>0.588</td>
<td>0.523</td>
<td>0.583</td>
<td>0.515</td>
<td>0.571</td>
<td>0.554</td>
</tr>
<tr>
<td>CCA</td>
<td>0.568</td>
<td>0.596</td>
<td>0.539</td>
<td>0.598</td>
<td>0.533</td>
<td>0.580</td>
<td>0.569</td>
</tr>
<tr>
<td>OT</td>
<td>0.581</td>
<td>0.613</td>
<td>0.556</td>
<td>0.600</td>
<td>0.544</td>
<td>0.582</td>
<td>0.580</td>
</tr>
<tr>
<td>RT</td>
<td>0.564</td>
<td>0.605</td>
<td>0.565</td>
<td>0.607</td>
<td>0.551</td>
<td>0.596</td>
<td>0.581</td>
</tr>
<tr>
<td>ORT</td>
<td>0.591</td>
<td>0.630</td>
<td>0.586</td>
<td>0.629</td>
<td>0.583</td>
<td>0.631</td>
<td>0.608</td>
</tr>
</tbody>
</table>

with SemEval baseline (Cer et al., 2017). Note that the EN-EN is equal to the one achieved by Brychcín (2020) and thus validates our implementation.

In Table 4 we can see the mean Pearson correlations over language pairs. The worst results were achieved on DE-Fr and CS-Fr which is not surprising as they are distant language families (Slavic-Romance and Germanic-Romance). In general, French appears to be the most difficult to understand the meaning compared to other language combinations in this dataset.

The linear mapping techniques are sorted by their performance e.g. OT outperforms LS and CCA. The best performing setting is Orthogonal Ranking Transformation and Optimal Matching with IDF weighting.

5 Conclusion and Future Work

In this paper we presented new STS datasets for both cross-lingual and monolingual STS and provided them to the research community. We extended experiments of previous work on STS using linear transformations to create cross-lingual semantic spaces, by conducting initial experiments on the newly created datasets. We confirmed the findings of Brychcin (2020) by replicating three (previously published) approaches to combine information from word representations.

The used STS system does not require sentence similarity supervision and the only cross-lingual information is a bilingual dictionary. In the future, we intend to investigate the use of unsupervised methods to create the bilingual dictionary.

Acknowledgements

This work has been partly supported from ERDF "Research and Development of Intelligent Components of Advanced Technologies for the Pilsen Metropolitan Area (InteCom)" (no.: CZ.02.1.01/0.0/0.0/17_048/0007267).

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei


Takashi Wada, Tomoharu Iwata, and Yuji Matsumoto. 2019. Unsupervised multilingual word embedding with limited resources using neural language models. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics,
Relation Extraction using Multiple Pre-Training Models in Biomedical Domain

Satoshi Hiai¹, Kazutaka Shimada¹, Taiki Watanabe², Akiva Miura², Tomoya Iwakura²
¹Kyushu Institute of Technology, Fukuoka, Japan
²Fujitsu Ltd., Tokyo, Japan
³Atid Ltd., Tokyo, Japan

s_hiai@pluto.ai.kyutech.ac.jp, shimada@ai.kyutech.ac.jp
{watanabe-taiki, iwakura.tomoya}@fujitsu.com
akiva.miura@studio-atid.com

Abstract

The number of biomedical documents is increasing rapidly. Accordingly, a demand for extracting knowledge from large-scale biomedical texts is also increasing. BERT-based models are known for their high performance in various tasks. However, it is often computationally expensive. A high-end GPU environment is not available in many situations. To attain both high accuracy and fast extraction speed, we propose combinations of simpler pre-trained models. Our method outperforms the latest state-of-the-art model and BERT-based models on the GAD corpus. In addition, our method shows approximately three times faster extraction speed than the BERT-based models on the ChemProt corpus and reduces the memory size to one sixth of the BERT ones.

1 Introduction

The amount of biomedical documents is increasing rapidly. The documents contain valuable knowledge, such as chemical compound names and their relations. However, the current knowledge extraction is considerably manual. Therefore, a demand for extracting knowledge automatically from large-scale biomedical text data is increasing.

Biomedical relation extraction (RE) models based on BERT (Devlin et al., 2019) have shown great performance (Lee et al., 2019; Beltagy et al., 2019). The methods using BERT models pre-trained on biomedical corpora achieved state-of-the-art (SOTA) performance on several biomedical RE datasets. However, BERT models require a huge amount of computational resources and generally need a long time for extraction processes. By processing data in parallel with multiple computational resources, we can process larger text as compared with a single resource. However, a high-end GPU or a distributed environment for efficient computation is not available in many situations. Even if we can utilize such computational resources, substantial energy consumption becomes a problem (Strubell et al., 2019). Therefore, more lightweight and accurate RE models are expected.

In this paper, we construct a biomedical RE model by combining word embeddings obtained from multiple lightweight models. The RE model can be executed in a wide range of environments, such as a CPU and a middle-class GPU, by reducing memory consumption during the learning process or the inference process. During the inference process of our proposed model, the amount of calculation can be suppressed and the model can process documents at high speed. Since the calculation in each lightweight model is small scale, memory consumption can be suppressed. Furthermore, by selecting whether or not to utilize each word embeddings, we can customize the model to be suitable for the computer environment. Hence, our model can process faster than the BERT-based models in the inference process.

We adopt more lightweight pre-trained models: ELMo (Peters et al., 2018) and Contextual String Embeddings (CSE) (Akbik et al., 2018). ELMo is contextualized word-level embeddings from the language model (LM) based on multiple layers of bidirectional long-short term memories (Bi-LSTMs) (Hochreiter and Schmidhuber, 1997). On the other hand, CSE is character-level embeddings of each input word from LM based on single-layer Bi-LSTM. Subword information of words plays an important role in the estimation of kinds and features of chemicals since chemical names tend to contain characteristic sub-word patterns such as prefixes and suffixes. Therefore, we propose RE models that combine ELMo and CSE to utilize both word-level and character-level features effectively.

We investigate the effectiveness of our RE mod-
els in terms of not only the accuracy but also the processing speed and the memory size. The contributions of this paper are as follows:

- We apply the strategy that feeds the complementary features from pre-training models to RE tasks in the biomedical domain: GloVe, ELmo, and CSE.
- The proposed model outperforms the latest SOTA F1 score on the GAD corpus. As a case study, we show that BERT-based models do not always produce the best performance.
- Our model performs approximately three times faster extraction speed than BERT-based models on the ChemProt corpus and reduces the memory size to one sixth of the BERT ones.

## 2 Related Work

Beltagy et al. (Beltagy et al., 2019) have proposed a method with pre-trained the BERT model called SciBERT. They pre-trained the BERT model on the large scale computer science and biomedical corpora. They constructed a new vocabulary of the BERT model for the tasks of science and biomedical domains. The SciBERT model with their vocabulary achieved SOTA performance on several benchmark tasks on the domains. Lee et al. (Lee et al., 2019) have proposed a model called BioBERT. They pre-trained the BERT model on a large scale biomedical corpus containing 4.5 billion words. They applied the model to biomedical NER, RE, and question answering tasks and achieved high performance on the benchmark tasks. For the RE tasks, they utilized the sentence classifier of the original version of BERT, which uses a [CLS] token for the classification of relations. They used pre-defined strings such as @GENE$ and @DISEASE$ to express a pair of target entities. For instance, a sentence with two target entities (gene and disease in this case) is represented as **Example 1**.

**Example 1** Serine at position 986 of @GENE$ may be an independent genetic predictor of angiographic @DISEASE$.

Zhou et al. (Zhou et al., 2016) have proposed a relation classification model with an attention-based Bi-LSTM model. They used pre-defined indicator tags to express a pair of target entities. For instance, a sentence with a pair of target entities is represented as **Example 2**.

**Example 2** <e1> Flowers </e1> are carried into the <e2> chapel </e2>.

Entity pairs were anonymized using the predefined strings in the method of Lee et al. (Lee et al., 2019). In contrast, this model can predict a relation class using surface information of the target entity pair. Sub-word information such as prefixes and suffixes plays an important role in estimations of kinds and features of chemicals. Therefore, we express the entity pair using tags in our method.

BERT is constructed by multiple layers of multi-head self-attention layers and requires large-scale computational resources. More lightweight pre-training LM models have also been proposed. Jin et al. (Jin et al., 2019) have proposed models for biomedical NLI tasks using ELMo pre-trained on large-scale in-domain text data. ELMo is an LM based on multi-layer Bi-LSTMs for aiming at obtaining contextualized word-level embeddings. CSE is generated by a character-level LM. The LM is lightweight since it is constructed with a single layer of Bi-LSTM. Sharma et al. (Sharma and Daniel Jr, 2019) have proposed a biomedical NER method with CSE generated from the LM pre-trained on a biomedical corpus. Watanabe et al. (Watanabe et al., 2019) have proposed a method with a multi-task learning model using CSE. Their method achieved SOTA performance on the biomedical NER task. Sharma et al. and Watanabe et al. evaluated the effectiveness of CSE on the biomedical NER tasks. However, they did not evaluate the effectiveness of CSE on the biomedical RE task. We evaluate the effectiveness of ELMo and CSE on the biomedical RE tasks.

## 3 Proposed Method

Figure 1 shows an overview of our method. We incorporate three types of word embeddings into the RE model: GloVe (Pennington et al., 2014),
CSE, and ELMo. First, we train a character-level LM for CSE and a word-level LM for ELMo using large-scale biomedical corpora. Then we obtain GloVe, CSE, and ELMo vectors corresponding to each word in a sentence as shown in the middle of Figure 1. Next, we construct an RE model as shown in the top of Figure 1. We explain the pre-training procedure for the GloVe embeddings and the language models for ELMo and CSE vectors in Section 3.1. Then, we explain the RE model based on the combinations of multiple word embeddings in Section 3.2.

### 3.1 Pre-Training

We obtain word embeddings by concatenating GloVe, CSE, and ELMo vectors for training and extraction. We use the GloVe embeddings trained on general domain corpora (the Wikipedia and the Gigaword corpus). For pre-training the CSE language model, we use the PubMed, the PMC, and the ChemRxiv datasets. The data from PubMed, PMC, and ChemRxiv contain 190K, 270K, and 300K biomedical papers, respectively. We use the ELMo embeddings trained on the PubMed corpus.

### 3.2 Relation Extraction Model

For the RE task, we apply a Bi-LSTM with an attention model (Zhou et al., 2016). The relation extraction model outputs a predicted class label. Here, we express the stacked embeddings as \( X = x_1, x_2, ..., x_n \). The predicted class label is computed as follows:

\[
\begin{align*}
\overrightarrow{h_i} &= LSTM(x_i, \overrightarrow{h}_{i-1}) \\
\overleftarrow{h_i} &= LSTM(x_i, \overleftarrow{h}_{i+1}) \\
h_i &= [\overrightarrow{h_i}; \overleftarrow{h_i}]
\end{align*}
\]

where \( x_i \) is the i-th input vector. \( \overrightarrow{h_i} \) and \( \overleftarrow{h_i} \) are hidden states of the forward LSTM and backward LSTM, respectively. \([::]\) indicates concatenation of two vectors. We calculate a weight \( a_i \) for each hidden state \( h_i \) as follows:

\[
\begin{align*}
m_i &= \omega^T \tanh(h_i) \\
a_i &= \frac{\exp(m_i)}{\sum_{j=1}^{n} \exp(m_j)}
\end{align*}
\]

where \( \omega \) is a vector of trainable parameters. We obtain the final hidden state \( h^* \) as follows:

\[
\begin{align*}
r &= \sum_{i=1}^{n} a_i h_i \\
h^* &= \tanh(r)
\end{align*}
\]

Then, the model calculates a predicted label \( \hat{y} \) as follows:

\[
\hat{y} = \arg \max_{y} P(y|X)
\]

During training, we use the loss function:

\[
Loss_{RE} = -\frac{1}{N} \sum_{i=1}^{N} \log(p(y_i|X))
\]

where \( N \) is the number of class labels.

We use an SGD optimizer (Bottou, 1991). We set parameters as follows: a learning rate is 0.1, a batch size is 32, and the number of hidden units is 256.

### 4 Experimental Settings

#### 4.1 Dataset

We use the Genetic Association Database (GAD) (Bravo et al., 2015) and the Bio-Creative VI Chemical-Protein RE dataset (ChemProt) (Islamaj Doğan et al., 2017) as RE datasets to evaluate our model. Table 1 shows the statistics for each dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Class</th>
<th># samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAD</td>
<td>Positive</td>
<td>2,801</td>
</tr>
<tr>
<td></td>
<td>Negative</td>
<td>2,529</td>
</tr>
<tr>
<td>ChemProt</td>
<td>Positive</td>
<td>1,973</td>
</tr>
<tr>
<td></td>
<td>Negative</td>
<td>5,002</td>
</tr>
<tr>
<td></td>
<td>CPR:3</td>
<td>471</td>
</tr>
<tr>
<td></td>
<td>CPR:4</td>
<td>726</td>
</tr>
<tr>
<td></td>
<td>CPR:6</td>
<td>1,814</td>
</tr>
<tr>
<td></td>
<td>CPR:9</td>
<td>31,298</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>9,986</td>
</tr>
</tbody>
</table>

Table 1: Dataset statistics. The number of samples for ChemProt is the sum of the number of samples in training, validation, and test sets.

The GAD dataset contains gene-disease relations. Relations between a gene and a disease within the same sentence were annotated. It is a binary classification task. The ChemProt dataset consists of 2,432 PubMed abstracts with chemical-protein relations annotated by domain experts. This

---

1https://nlp.stanford.edu/projects/glove/
3https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/
4https://chemrxiv.org/
5https://allennlp.org/elmo
Table 2: Experimental results. Bold and underline indicate the best score and the second best score, respectively. The values of E-SVM (Bhasuran and Natarajan, 2018), SPINN (Lim and Kang, 2018), and BioBERT (Lee et al., 2019) are referred from the original papers. For SciBERT (Beltagy et al., 2019), because the experimental setting in the paper differs from the setting of BioBERT, we re-evaluate it with the same setting as BioBERT.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>E-SVM</th>
<th>SPINN</th>
<th>SciBERT</th>
<th>BioBERT</th>
<th>Baseline (GloVe)</th>
<th>Proposed1 (GloVe+CSE)</th>
<th>Proposed2 (GloVe+ELMo)</th>
<th>Proposed3 (GloVe+CSE+ELMo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAD</td>
<td>79.21</td>
<td>-</td>
<td>85.54</td>
<td>76.43</td>
<td>77.31</td>
<td>78.74</td>
<td>83.10</td>
<td>82.64</td>
</tr>
<tr>
<td>ChemProt</td>
<td>89.25</td>
<td>-</td>
<td>80.61</td>
<td>87.65</td>
<td>80.01</td>
<td>87.43</td>
<td>85.43</td>
<td>86.36</td>
</tr>
<tr>
<td>F</td>
<td>83.93</td>
<td>-</td>
<td>82.83</td>
<td>81.61</td>
<td>78.59</td>
<td>82.83</td>
<td>84.16</td>
<td>84.38</td>
</tr>
<tr>
<td>P</td>
<td>-</td>
<td>74.85</td>
<td>79.73</td>
<td>77.02</td>
<td>58.13</td>
<td>67.41</td>
<td>76.49</td>
<td>76.05</td>
</tr>
<tr>
<td>R</td>
<td>56.06</td>
<td>70.85</td>
<td>75.90</td>
<td>57.43</td>
<td>58.79</td>
<td>65.82</td>
<td>66.17</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>64.11</td>
<td>75.03</td>
<td>76.46</td>
<td>57.78</td>
<td>62.81</td>
<td>70.76</td>
<td>70.77</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 shows the experimental results. For all the datasets, we reported precision, recall, and F-measure (F1) scores for the positive classes.

5 Experimental Results

5.1 Evaluation on accuracy

The F1 score of the Proposed1 (GloVe+CSE) outperformed the F1 score of Baseline (GloVe) for GAD and ChemProt datasets. The F1 score of the Proposed3 (GloVe+CSE+ELMo) outperformed Baseline and other all the proposed methods. The
Table 3: Extraction time required for the test-set of ChemProt.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (s)</th>
<th>Sample (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed1 (GloVe+CSE)</td>
<td>86.53</td>
<td>5.99</td>
</tr>
<tr>
<td>Proposed2 (GloVe+ELMo)</td>
<td>212.51</td>
<td>14.71</td>
</tr>
<tr>
<td>Proposed3 (GloVe+CSE+ELMo)</td>
<td>239.06</td>
<td>16.55</td>
</tr>
<tr>
<td>SciBERT</td>
<td>537.32</td>
<td>37.20</td>
</tr>
<tr>
<td>BioBERT</td>
<td>680.39</td>
<td>47.11</td>
</tr>
</tbody>
</table>

result shows the effectiveness of LMs pre-trained on the biomedical corpora.

For the GAD dataset, the Proposed3 showed the best performance. It outperformed SciBERT, BioBERT, and E-SVM. For the ChemProt dataset, although this method also outperformed SPINN, the previous SOTA method, it did not reach the scores of SciBERT and BioBERT.

Another approach to improve the accuracy is to incorporate a BERT model with our combination models. We also attempted to incorporate the BERT embeddings generated by the BioBERT model into our model although the result is not documented in Table 2. However, all the methods combining BERT embeddings (GloVe+BERT, GloVe+ELMo+BERT, GloVe+CSE+BERT, and GloVe+ELMo+CSE+BERT) resulted slightly lower performance than BioBERT. In addition, the models with BERT embeddings lead to vanishment of the effectiveness and motivation, namely construction of a lightweight model.

### 5.2 Evaluation on Processing Speed

Table 3 shows the extraction time of the proposed methods, SciBERT, and BioBERT for processing the test set of ChemProt. The F1 score of the Proposed3 was approximately 6 points lower than BioBERT. However, the extraction speed was 2.85 times faster than that of BioBERT. Although the F1 score of the Proposed1 was approximately 8 points lower than the Proposed3, the extraction speed was 2.76 times faster than that of the Proposed3. We can see a trade-off between the F1 scores and the extraction speed. If users need the extraction speed for the application, our method is useful although the accuracy is comparatively sacrificed.

### 5.3 Evaluation on Memory Size

In the experiment, we used NVIDIA Tesla V100 GPU with 32GB memory. This is the high-end GPU for data center use at present. We observed the maximum memory consumption during the learning execution of BioBERT and the proposed method. BioBERT consumed approximately 12GB of memory. It indicates that BioBERT needs high-end GPUs to execute the learning. On the other hand, our method consumed approximately 2GB of memory. The memory consumption of our method was lower than that of BioBERT (1/6). In addition, considering the memory consumption, we believe that our model can be executed even on a middle-class GPU.

### 6 Discussion

We showed the effectiveness of our method in the previous section. In this section, we discuss our method from various perspectives, including some negative results. First, we discuss a comparison between the BERT-based models and the proposed method in Section 6.1. Then, we discuss the experimental settings of the proposed method in Section 6.2 and 6.3.

#### 6.1 Datasets and Model Performance

For the GAD dataset, our method outperformed the BERT-based models. However, for the ChemProt dataset, the precision, recall, and F1 scores were lower than the scores of the BERT-based models. In the GAD dataset, the number of the negative samples is almost the same as the number of positive samples as shown in Table 2. On the other hand, in the ChemProt dataset, the number of negative samples is three times more than the number of positive samples. We analyzed the classification errors of our method for the positive samples of the ChemProt dataset. About 92% of the misclassifications were positive samples classified into the negative class, not into the other positive classes. It seems that the data imbalance affects our model.

---

*We did not evaluate the extraction time for the GAD dataset because the dataset size is small. However, it seems that there is no large difference between the average time per sample for the GAD and the time for the ChemProt.*
Table 4: Effects of the pre-training corpus. The score of PubMed+PMC+ChemRxiv is the same as the Proposed3 in Table 2.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>PubMed</td>
<td>69.57</td>
</tr>
<tr>
<td>PubMed+ChemRxiv</td>
<td>69.31</td>
</tr>
<tr>
<td>PubMed+PMC</td>
<td>70.19</td>
</tr>
<tr>
<td>PubMed+PMC+ChemRxiv</td>
<td>70.77</td>
</tr>
</tbody>
</table>

Table 5: Effects of the target entity pair indicators.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Tag Replacement</th>
<th>Proposed 3 (GloVe+CSE+ELMo)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>P 82.64</td>
</tr>
<tr>
<td>GAD</td>
<td></td>
<td>R 86.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F 84.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P 76.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R 66.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F 70.77</td>
</tr>
</tbody>
</table>

6.2 Effects of Pre-Training Corpora

We used the CSE pre-trained on the PubMed, PMC, and ChemRxiv corpora in Proposed3. We evaluated the effectiveness of each corpus for the pre-training. Table 4 shows the results for the ChemProt. The F1 score of the model pre-trained on the PubMed + ChemRxiv was lower than the model pre-trained on the PubMed. On the other hand, the F1 score of the model pre-trained on the PubMed + PMC was higher than the model pre-trained on the PubMed. The F1 score of the model pre-trained on the PubMed + PMC + ChemRxiv was the best score. The combination of PubMed, PMC, and ChemRxiv was effective although introducing ChemRxiv alone provided no benefit to the pre-training.

6.3 Effects of Target Entity Pair Indicators

We used the indicator tags to express a pair of two target entities. On the other hand, Lee et al. (Lee et al., 2019) replaced the target entity pair with pre-defined strings. In this section, we compare the methods using the indicator tags with the methods using the entity pair replacement. For the GAD dataset, we replaced diseases and genes with pre-defined strings @DISEASE$ and @GENES$, respectively. For the ChemProt dataset, Lim and Kang (2018), replaced chemicals and proteins with pre-defined strings “bc6entc” and “bc6entg”, respectively. We used the same pre-defined strings that were used in Lim and Kang.

For instance, 1-aminoadamantane and Fos in Example 4 are replaced with “bc6entc” and “bc6entg” respectively as shown in Example 5.

Example 4 Amantadine (1-aminoadamantane) induced Fos expression in the central, dorsal-medial and ventral-medial part of the striatum.

Example 5 Amantadine (bc6entc) induced bc6entg expression in the central, dorsal-medial and ventral-medial part of the striatum.

We evaluated the proposed3 using each of the indicator tags and the entity pair replacement. Table 5 shows the experimental results for the GAD and ChemProt datasets. As a result, the use of the indicator tags was effective as compared with that of the replacement approach. We can use the surface information of the target entity pair by using indicator tags. Therefore, the result shows the effectiveness of the surface information of the entity pairs.

7 Conclusions

In this paper, we reported the effectiveness of lightweight and high-performance RE models for the biomedical domain. Our method used the combination of word embeddings generated by the pre-trained LMs (the ELMo model and the CSE model). The ELMo model is a word-level LM and the CSE model is a character-level LM. We proposed RE models that combined ELMo and CSE to utilize
both word-level and character-level features effectively.

We evaluated the proposed methods on the biomedical RE datasets. We compared our methods with BERT-based methods (BioBERT and SciBERT) and the SOTA methods. We also evaluated the model performance and the inference time. Experimental results showed the effectiveness of the combinations of the LMs. For the GAD dataset, we obtained the SOTA score. For the ChemProt dataset, our model showed approximately three times faster extraction speed than BioBERT. In addition, our model reduced the memory size to one sixth of the BERT-based models. However, the F1 score of our model was lower than that of BioBERT for the ChemProt dataset. In future work, we analyze the causes of the high extraction speed and the low performance of our model for the ChemProt dataset in terms of the parameter size and architectures.

Acknowledgment

A part of this work was carried out in the RIKEN Center for Advanced Intelligence Project.

References


Álex Bravo, Janet Piñero, Núria Queralt-Rosinach, Michael Rautschka, and Laura I. Furlong. 2015. Extraction of relations between genes and diseases from text and large-scale data analysis: implications for translational research. BMC Bioinformatics, 16:55.


Rezarta Islamaj Doğan, Sun Kim, Andrew Chatr-aryamontri, Chih-Hsuan Wei, Donald C Comeau, Rui Antunes, Sérgio Matos, Qingyu Chen, Aparna Elangoan, Nagesh C Panyam, Karin Verspoor, Hong-fang Liu, Yanshan Wang, Zhuang Liu, Berna Altunel, Zehra Melce Hüsünbeyi, Arzuca Özgür, Aris Fergadis, Chen-Kai Wang, Hong-Jie Dai, Tung Tran, Ramakanth Kavuluru, Ling Luo, Albert Steppi, Jinfeng Zhang, Jinchan Qu, and Zhiyong Lu. 2017. Overview of the bioCreative vi precision medicine track: mining protein interactions and mutations for precision medicine. In the sixth BioCreative challenge evaluation workshop, volume 1, pages 141–146.


Discussion Structure Prediction Based on a Two-step Method

Takumi Himeno and Kazutaka Shimada
Department of Artificial Intelligence, Kyushu Institute of Technology, Japan
t_himeno@pluto.ai.kyutech.ac.jp, shimada@ai.kyutech.ac.jp

Abstract
Conversations are often held in laboratories and companies. A summary is vital to grasp the content of a discussion for people who did not attend the discussion. If the summary is illustrated as an argument structure, it is helpful to grasp the discussion’s essentials immediately. Our purpose in this paper is to predict a link structure between nodes that consist of utterances in a conversation: classification of each node pair into “linked” or “not-linked.” One approach to predict the structure is to utilize machine learning models. However, the result tends to over-generate links of nodes. To solve this problem, we introduce a two-step method to the structure prediction task. We utilize a machine learning-based approach as the first step: a link prediction task. Then, we apply a score-based approach as the second step: a link selection task. Our two-step methods dramatically improved the accuracy as compared with one-step methods based on SVM and BERT.

1 Introduction
Meetings are often held in laboratories and companies to come up with new research ideas and management strategies. A summary is vital to grasp the content of a discussion for people who did not attend the discussion. Summaries are suitable for understanding the main points in discussions. Assume that a summary is illustrated as a discussion structure. The summary is more powerful and helpful to understand the main points in the discussion because users can immediately capture the flow of the discussion by using links between utterances. For this purpose, we need to predict the discussion structure of each discussion.

Argument mining is one of the tasks to construct a structure of sentences (Stab and Gurevych, 2017a). It automatically derives the structure of argumentation from unstructured documents such as essays. It consists of four subtasks as follows: component identification, component classification, relation identification, and relation classification. Component identification is a task that extracts argument components from a given document. Argument components denote sentences and paragraphs related to the discussion structure. Component classification is a task that assigns a label, e.g., claim, to each argument component. Relation identification is a task that predicts whether each pair of argument components is related or not. Relation classification is a task that assigns a label, such as “attack” and “support,” to the related pairs of argument components.

In this paper, we deal with relation identification for constructing a discussion structure in a multi-party conversation. In other words, we construct a link prediction model for nodes consisting of some utterances. Methods in previous work often predicted the discussion structures by using machine learning approaches, such as neural networks. Himeno and Shimada (2020) have reported that such machine learning models tended to over-generate links between nodes. Here we focus on some rules in discussion structures; e.g., a child node has one parent note. We incorporate score-based selection rules with the machine learning model as post-processing to improve the accuracy. We introduce top-down and bottom-up approaches for selecting edges. The result shows that the proposed methods are more accurate than the method without the selection.

2 Related Work
In recent years, argument mining is attracting attention in natural language processing. Argument mining is a task to construct the structure of a document. It is applied to many natural language processing tasks such as document summarization (Barker
and Gaizauskas, 2016; Peldszus, 2014), the automatic scoring of essays (Ghosh et al., 2016), the paper writing support (Stab and Gurevych, 2017b; Nguyen and Litman, 2016), the information retrieval (Stab et al., 2018) and so on. Stab and Gurevych (2014) have tackled the relation identification for essays written by students. They created some features capturing the characteristics of the essay and then predicted links between argument components. The essay is usually formalized, such as the form of a claim followed by premises. However, multi-party conversations are not always formalized because many people freely speak to assert their opinions.

Discussion structures can be regarded as a kind of graph structure. In fact, some studies handled the selection of edges of argument structures as the shortest path problem of graphs (Dijkstra et al., 1959; Gabow and Tarjan, 1985). The shortest path problem is an optimization problem to find the path with the minimum weight among the paths connecting two given nodes in a weighted graph. In the shortest path problem, the method usually optimizes the entire path from parent nodes to multiple terminal nodes. We also apply this idea to our task, namely relation identification. However, it is not always suitable to adapt our task directly because there are some conditions in discussions: e.g., only one parent node for a child node. In this paper, we propose a link selection method for local utterance pairs to handle the characteristics of the discussions.

In addition, many studies have focused on the visualization of discussions (Chamberlain et al., 2018; Lugini et al., 2020). In this paper, we also visualize predicted links to understand the result easily.

### 3 Dataset and Task

#### 3.1 Dataset

In this paper, we use the AMI corpus, a multi-party conversation corpus (Carletta et al., 2005). It contains various useful annotations, such as the argument structure and time information, to predict a link between nodes. Each node consists of one or more utterances. We use scenario meetings that are held with the discussion points given in advance. In the discussion setting, four employees in different roles in a company discuss developing a new TV remote control that replaces an old-style TV remote control for consumers on the market. One discussion is held four times. Each utterance in the AMI corpus contains speaker ID, time information, and a dialog act.

In this paper, we use the annotated data based on the Twente argument schema (TAS) to contain the link between nodes (Rienks et al., 2005). TAS is an annotation schema created to clarify the discussion structure which arises from the scenario meeting of the AMI corpus. The discussion structure in TAS consists of two elements. One is a node, and the other is an edge. The node in TAS contains parts of, or even complete, speaker turns. The edge in TAS represents the type of relation between the nodes. In TAS, unit labels that represent the role of the node are also annotated. The details of the unit labels are shown in Table 1. In addition, TAS defines “discussion” as segments in the meeting (“Dialog”). One dialog consists of one or more discussions. One discussion consists of some nodes. One unit label is assigned to each node.

#### 3.2 Task

Figure 1 shows an example of relation identification in this paper. In Figure 1, the dialog contains two discussions: discussion1 and discussion2. The two discussions contain three nodes and two nodes,

![Figure 1: Relation identification. We handle all combinations of node pairs in each discussion for the relation identification task. The task is to classify each pair into “linked” or “not-linked.”](http://groups.inf.ed.ac.uk/ami/download/)

<table>
<thead>
<tr>
<th>Tag Type</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statement</td>
<td>A claim without a weakening qualifier</td>
</tr>
<tr>
<td>Weak Statement</td>
<td>A claim with a weakening qualifier</td>
</tr>
<tr>
<td>Open Issue</td>
<td>An issue that is raised where every possible response could be a solution</td>
</tr>
<tr>
<td>A/B Issue</td>
<td>An issue that is raised where the possible responses are explicitly enumerated</td>
</tr>
<tr>
<td>Yes/No Issue</td>
<td>An issue that is raised where the possible responses are Yes and No</td>
</tr>
<tr>
<td>Other</td>
<td>Not fitting any of the other Unit Labels</td>
</tr>
</tbody>
</table>

Table 1: Detail of the unit labels in the TAS.
Black is better, I think. It doesn’t show dirt. Definitely! It’s a popular color. I don’t like black color.

A: Black is better, I think.
B: It doesn’t show dirt.
C: It’s a popular color.
D: Definitely!
B: I don’t like black color.

4 Proposed Method

In this section, we explain our method with two steps: link prediction and link selection. Figure 2 shows an overview of our method. The first step (link prediction) is based on machine learning techniques. We compare two models; one is Support Vector Machines (Vapnik, 2000) with word embeddings and selected features, and the other is BERT (Devlin et al., 2019). However, the models tend to over-generate links between nodes in this task. Hence, we introduce link selection approaches as the second step: top-down and bottom-up approaches.

4.1 Link prediction

4.1.1 SVM

We apply Support Vector Machines (SVMs) to the link prediction task. Figure 3 shows the outline of the SVM-based model. It learns the link prediction model with two embeddings of two nodes and eight features. As the word embedding, we use word2vec (W2V) published by Google. We generate the vector space as follows:

\[ V_{node_i} = \sum_{x=1}^{m} v_x \]  

where \( v_x \) denotes the word vectors of \( node_n \) and \( m \) denotes the size of the \( node_n \). For example, assume that we predict the relation between \( node_i \) and \( node_j \) consisting of some word embeddings \( (v_x) \). We obtain two summed word embedding vectors, namely \( V_{node_i} \) and \( V_{node_j} \) from \( node_i \) and \( node_j \). Finally, SVMs learn and predict the relation by using concatenated \( V_{node_i} \) and \( V_{node_j} \).

We also utilize eight features as follows:

- Number of words in node pair
  - If a speaker supports and attacks another speaker’s claim, the size of the node tends to be larger. In a similar way, the node also tends to be larger if the speaker wants to convey much information to the other speakers.

Figure 3: The link prediction model with SVM.
On the other hand, the size of a node becomes smaller if the node consists of short utterances, such as back-channel feedback. Thus, the size of each node is one of the important characteristics. To capture this feature, we use the number of words in each node.

- **Number of common words in node pairs**
  If two nodes are related to a common topic, words in them are frequently overlapped. Therefore, we count the number of common words that appear in each node as the feature.

- **Speaker information**
  Agreement or negative statements to an opinion from a speaker tends to be uttered from another speaker. Besides, the situation that the same speaker gives a positive opinion to his/her own claim or points out a problem of his/her claim is very rare. Therefore, the speaker information of each node has an important role in the relation between two nodes. We use the speaker ID of each node as the feature.

- **Time information**
  If the discussion is active, the time interval between nodes tends to become shorter. As another example, a link between a node in the early stage and a node in the last stage in a discussion is rare. In other words, far-flung nodes usually do not possess a link. Thus, time information between nodes has an important role. To capture this feature, we focus on time information in the corpus. We compute the time information by using the end time of node \( i \) and the start time of node \( j \) as the feature.

- **Distance between nodes**
  Assume that the discussion is stagnant. In this situation, the distance between nodes becomes short because the number of nodes in the stagnant situation becomes small\(^5\). Thus, the distance, namely the number of nodes between two nodes, is one important feature. Therefore, we sort the nodes in a discussion in terms of the timestamps and use the distance between nodes as the feature.

- **Dialog act**
  Dialog act tags are important information for the prediction model. For example, if a node contains an “Inform” tag, the node tends to connect with nodes containing “Backchannel” and “Assess” because of the nature of discussions. On the other hand, a node with an “Elicit-inform” tag does not usually connect with an “Inform” tag because the “Elicit-inform” tag is used by a speaker to request that someone else give some information while the “Inform” tag is used by a speaker to give information. Therefore, we use the distribution of 15 types of dialog acts in each node as the feature.

- **Unit label**
  The unit labels described in Section 3.1 also have an important role in the prediction of the link between nodes. They contain three types of labels related to questions: “Open Issue,” “A/B Issue,” and “Yes/No Issue.” If a node contains such tags, the node tends to connect with nodes that express positive/negative opinions. Besides, nodes with such tags do not generally connect with nodes about questions because it is a question-question pair. Therefore, we use the unit label of each node as the feature.

- **Polarity of node pair**
  Emotional information is also one of the characteristics of conversations. For example, a speaker may emotionally argue while claiming his/her opinion in a discussion. In a similar way, when a speaker may emotionally argue when he/she agrees or disagrees with another speaker’s question. To capture the information, we use Stanford CoreNLP (Manning et al., 2014). We compute the score (1 to 5) of each utterance by using CoreNLP. Then we compute the average score from the score of the utterances in each node. We use the average polarity score of each node as the feature.

### 4.1.2 BERT

The second model is BERT (Devlin et al., 2019). BERT is a Transformer-based machine learning model that is pre-trained by a large corpus. It can fine-tune the target tasks. BERT is known to perform well in various natural language processing.
4.2 Link selection

The link prediction models in Section 4.1 tend to over-generate links. In this section, we introduce two types of selection approaches: bottom-up and top-down.

4.2.1 Bottom-up approach

In this approach, first of all, each model (SVM and BERT) predicts the relation of each node: “link” or “not-link.” Figure 5 shows an overview of the bottom-up approach. In this figure, we obtain the link prediction result; e.g., node1 is linked to node2, node3, and node4. Then, we prune links on the basis of a cost parameter in the case that a child node has two or more parent nodes; e.g., node5 has two parents (node2 and node3). We employ the value of decision_function on scikit-learn for SVM. For BERT, we employ the value of the softmax function.

We select the node pair with the highest value as the final result if a child node has some parent nodes. In Figure 5, node5 has two parents with values (node2 with 0.8 and node3 with 0.9). The bottom-up approach selects node5 as the final link. We apply this bottom-up approach to the SVM model in Section 4.1.1 and the BERT model in Section 4.1.2 as the post-processing step, namely the selection process.

4.2.2 Top-down approach

Figure 6 shows an overview of the top-down approach. Firstly, in this approach, we assume that the first node in each discussion is the first parent node. Then, we predict the presence of the link between the parent node and each node in a discussion. In Figure 6, the solid line denotes that a link prediction model judged “the two nodes contain the link.” On the other hand, the dashed line denotes that the model judged “no link between the two nodes.” For example, the model judges that node3 and node5 are linked with node1 (the first parent node). Next, we set new parents by using the result; i.e., node3 and node5 are new parents. We repeat the process for the new parents. We select the final link in the case that two or more parent nodes have one child node. In Figure 6, node4 has two parents, namely node3 and node5. In other words, the model predicts that node3 is linked to node4 and node5 is also linked to node4. In a similar way with the bottom-up approach, we select the node pair with the highest value as the final result if a child node has some parent nodes: e.g., node5 in Figure 6.

The top-down approach needs a higher computa-
Table 2: Distribution of the experimental data. For the training data, we select 3850 not-linked pairs randomly to generate balanced training data.

<table>
<thead>
<tr>
<th></th>
<th>Dialog</th>
<th>Discussion</th>
<th>Linked</th>
<th>Not-linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>84</td>
<td>201</td>
<td>3850</td>
<td>38530</td>
</tr>
<tr>
<td>Dev</td>
<td>4</td>
<td>13</td>
<td>235</td>
<td>1822</td>
</tr>
<tr>
<td>Test</td>
<td>4</td>
<td>12</td>
<td>238</td>
<td>1875</td>
</tr>
</tbody>
</table>

Table 3: Results of SVM.

<table>
<thead>
<tr>
<th>Model</th>
<th>Link</th>
<th>Not-Link</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>SVM</td>
<td>0.38</td>
<td>0.84</td>
</tr>
<tr>
<td>+Bup</td>
<td>0.68</td>
<td>0.63</td>
</tr>
<tr>
<td>+Tdown</td>
<td>0.58</td>
<td>0.47</td>
</tr>
</tbody>
</table>

Table 4: Results of BERT.

<table>
<thead>
<tr>
<th>Model</th>
<th>Link</th>
<th>Not-Link</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>BERT</td>
<td>0.15</td>
<td>0.65</td>
</tr>
<tr>
<td>+Bup</td>
<td>0.40</td>
<td>0.31</td>
</tr>
</tbody>
</table>

5 Experiment

5.1 Experimental Settings

For the SVM model, we use LiBSVM (scikit-learn) for the implementation (Chang and Lin, 2011). The kernel function was the RBF function, and the cost parameter was 100. The other parameters were default values on scikit-learn. The setting was determined from the training data with a grid search on scikit-learn.

For the BERT model, we used the BERT-Base as the pre-trained model. The text has been lowercased. The batch size for the training was 16, and the batch size for the development and test was 8. The number of epochs was 3. We used cross-entropy as the loss function. The optimizer was Adam, and the learning rate was 0.00002.

We used 219 discussions from 92 dialogs of the AMI corpus. In this experiment, all nodes in each discussion were given, and we used oracle unit labels in the corpus for the feature extraction. We divided the AMI corpus into 201 discussions from 84 dialogs for the training data, 13 discussions from 4 dialogs for the development data, and 12 discussions from 4 dialogs for the evaluation data. As explained in Section 3.2, we generated all combinations of two nodes in each discussion. The distribution, such as the number of linked pairs and not-linked pairs, was shown in Table 2.

Table 2 said that the experimental data were imbalanced; the linked pairs were just 3850 as against 38530 not-linked pairs. Models generated from imbalanced data tend to become a weak classifier. Therefore, we reduced the imbalance of the training data. For all models, we randomly selected 3850 not-linked pairs from the original training data. Then, we generated each model, namely SVM and BERT, from the downsized and balanced training data.

5.2 Experimental Results

We compared the effectiveness of our two-step methods. Table 3 and Table 4 show the experimental result of SVMs and BERT, respectively. $T_{down}$ and $B_{up}$ denote the top-down and bottom-up approaches for each model, respectively. The boldface denotes the best score for each criterion, namely Precision, Recall, and F1-score, in the table. On the F1-score, the two-step methods with the bottom-up outperformed the methods without the link selection. The top-down approach on SVM improved the precision rate although the F1-score was lower than $SVM$ without selection. These results show the effectiveness of our two-step methods, namely the introduction of the link selection approach.

5.3 Discussion

The link prediction models obtained high recall rates for the class “link” (0.84 on SVM and 0.65 on BERT) while the precision rates were low. In other words, the outputs contained many mistakes: child nodes with some parent nodes. The problem was remedied by introducing link selection models, especially the bottom-up. On the other hand, the improvement of the top-down approach was limited. The reason is that the top-down approach sequentially predicts a link of two nodes and selects the link from the top node. Although it can hold the relation between two nodes, it is not suitable to hold the relation of the whole discussion. As a result, the result was not sufficient. Moreover, in the top-down approach, the mistakes of the link prediction in the early stage lead to a negative impact on the later stage accumulatively. To obtain higher
accuracy by the top-down approach, we need additional rules and conditions in the link selection process: e.g., appearance order in the discussion.

We visualized discussion structures from the predicted results. Figure 7 shows an example of the correct discussion structure of a discussion. Figure 8 and Figure 9 show the visualized structures of SVM and SVM with the bottom-up approach, respectively. Figure 10 and Figure 11 also show the visualized structures of BERT only and BERT with the bottom-up approach, respectively. In each figure, solid blue lines denote correct links, and dashed red lines denote incorrect links. From the figure, our method dramatically improved the discussion structure prediction task. In particular, the output of the BERT model was obviously refined, namely the deletion of dashed red lines, because the precision rate of the original BERT model was extremely low. For the SVM model, the output from our method reduced mis-prediction as a whole.

The parameters of this experiment were determined from the training data. Therefore, the parameters are not always the best parameters in the experiment. Moreover, we just evaluated our methods with one setting in Table 2. We need to investigate the best parameters and the effectiveness of our method through cross-validation. In addition, we used word2vec for the embeddings for SVM although we can currently obtain embeddings from BERT as a stronger embedding. The BERT embeddings might lead to the improvement of the accuracy of the SVM-based model. The replacement of word embeddings for SVMs is one future work.

In our previous work (Himeno and Shimada, 2020), link prediction models tended to over-generate links between nodes. In this experiment, the bottom-up approach was effective for both SVM and BERT. Therefore, we believe that our two-step method is versatile and effective for relation identification tasks of argument mining. However, we evaluated our method with only the AMI corpus. Applying our method to other corpora and evaluating the effectiveness of our method in relation identification tasks are important future work.

6 Conclusions

In this paper, we proposed two types of discussion structure prediction methods. They were based on a two-step architecture: link prediction and link selection. For the link prediction, we evaluated two machine learning models, namely SVM and BERT. These models tended to over-generate links be-
between nodes. To solve this problem, we introduced top-down and bottom-up approaches to the link selection task. Our methods outperformed SVM and BERT without the link selection approaches (0.53 vs. 0.66 on F1 for SVM and 0.24 vs. 0.41 on F1 for BERT). In the experiment, the bottom-up approach was better than the top-down approach. We visualized the discussion structures from the outputs. From the visualized data, we can see qualitatively that our methods dramatically improved the discussion structure prediction.

To obtain higher accuracy, the recall rate of the link prediction model is the most important factor. It was indicated by the results of SVM: the higher recall rate led to the best performance, as compared with the BERT-based model. Therefore the important future work is to improve the link prediction models by machine learning, especially the recall rate.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 20K12110.

References


On the Usefulness of Personality Traits in Opinion-oriented Tasks

Marjan Hosseinia
University of Houston
{ma.hosseinia}@edragut.temple.edu

Eduard Dragut
Temple University

dainis.boumer

Arjun Mukherjee
University of Houston
arjun@cs.uh.edu

Dainis Boumber
University of Houston

Abstract

We use a deep bidirectional transformer to extract the Myers-Briggs personality type from user-generated data in a multi-label and multiclass classification setting. Our dataset is large and made up of three available personality datasets of various social media platforms including Reddit, Twitter, and Personality Cafe forum. We induce personality embeddings from our transformer-based model and investigate if they can be used for downstream text classification tasks. Experimental evidence shows that personality embeddings are effective in three classification tasks including authorship verification, stance, and hyperpartisan detection. We also provide novel and interpretable analysis for the third task: hyperpartisan news classification.

1 Introduction

The vocabulary we use in everyday language is a rich source of information about our beliefs, thoughts, and personalities (Pennebaker et al., 2015). Many efforts in text analysis provide compelling evidence that our everyday language carries psychological cues (Gottschalk and Gleser, 1979; Stone et al., 1966; Weintraub, 1989; Pennebaker et al., 2015). With this study, we seek to determine the personality of a given text’s author as defined by the Myers-Briggs Type Indicators or MBTI (Myers and Myers, 1995). Myers-Briggs uses four binary dimensions to classify people (Introvert–Extrovert, Intuitive–Sensing, Thinking–Feeling, Judging–Perceiving), which gives 16 different types, such as INTJ and ENTJ. This work uncovers novel insights into the personality space of authors from their online writings.

The personality signal carries the fingerprint of the individual’s psyche and, even though noisy, can be useful (as shown in this work) for a variety of downstream NLP tasks, such as authorship verification, stance, and hyperpartisan detection. Personality prediction does not only benefit commercial applications and psychology but also is advantageous in health care. Recent works link personality types and social media behavior with depression and posttraumatic stress disorder (Preotjiuc-Pietro et al., 2015). This is significant because it opens new avenues for prevention care as certain personality types can anticipate mental illness and schizophrenia. (Mitchell et al., 2015).

This problem poses a non-trivial challenge because a good solution must be capable of capturing the complexity and the depth of the human psyche as expressed through text. Anything short of that will result in task-specific pattern-matching. It follows that the main technical difficulty presented by the task at hand is the discrepancy between the corpora concerning their distributions, which results in the domain shift. This is where our work becomes relevant as it aims to bridge the gap by transfer learning and universal language understanding.

The problem is also challenging because the human psyche is complex in its nature. The labels are fuzzy as the label distribution changes from population to population and the ground truth is not derived by an objective method; rather, it is a set of ideas generally agreed upon by specialists and society. Furthermore, high quality curated datasets constructed by professional psychologists are difficult to obtain due to privacy reasons.

Models have only recently reached the capacity required. Our approach uses transfer learning through language understanding for personality prediction. We create a unified dataset from the collection of user inputs of three available MBTI datasets (Gjurković and Šnajder, 2018; Mitchell, 2017; Plank and Hovy, 2015) originating from social media platforms including Reddit, Twitter, and Personality Cafe forum. We investigate how transfer learning with pretrained transformers con-
tributes to personality prediction under a multi-label multi-class classification strategy. We analyze the relationship between personality types with the three specific tasks of stance, authorship, and hyperpartisan news classification. The results on the unified dataset show that transfer learning along with pretrained bidirectional transformer models effectively changes the Hamming loss, F1, and Jaccard similarity for multi-label personality prediction. The contributions of our paper are listed below:

- We propose to use the flow of sentiments across a document as a proxy for Myer-Briggs personality type and use a transformer-based model to predict personality type.
- We show the usefulness of personality traits on three downstream text classification tasks: authorship verification, stance, and hyperpartisan detection. The technical novelty is on transfer learning of our pretrained personality model to improve NLP downstream tasks.
- We give an in-depth statistical analysis of the effect of using personality information in the task of hyperpartisan news classification.

In the following sections we introduce and evaluate the personality model, then analyze its application in the three text classification tasks (mentioned above) using transfer learning.

2 Related Work

Personality prediction from text is a challenging task (Štajner and Yenikent, 2021; Yang et al., 2020) and many personality prediction approaches rely on crafted features which can range from simple ones, such as TF-IDF of word or character n-grams to the ones produced by tools such as Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2015), which extracts anything from low-level information such as Part Of Speech tags and topical preferences to psychological categories. These features are often supported by various psycholinguistic word lists that aim to detect emotions and sensory experiences (Preoțiuc-Pietro et al., 2017).

Our work uses a bidirectional transformer to predict MBTI personality types using a large collection of data obtained from three existing personality datasets. Utilization of the pretrained word embeddings (Mikolov et al., 2013; Pennington et al., 2014) in many deep learning models indicates that leveraging knowledge obtained from unsupervised learning boosts the performance. Recently, language models pretrained on a large amount of raw text were shown to provide representations applicable to a wide variety of tasks with minimal fine-tuning (Radford et al., 2018; Howard and Ruder, 2018; Peters et al., 2018a). These models can be effectively generalized to many downstream tasks and adapted to different domains. Below are the three representative studies on utilizing online user-generated text for personality prediction. They are annotated with self-reported MBTI personality types of users.

Reddit9K dataset is a large-scale dataset constructed from the posts and comments of 9K Reddit users. It is labeled with MBTI indicators and covers a wide variety of topics (Gjurković and Šnaidjer, 2018). The authors extract user activity and linguistic features including word and character n-grams, LIWC word categories (Pennebaker et al., 2015), and two Psycholinguistic dictionaries (Preoțiuc-Pietro et al., 2017; Coltheart, 1981). Support Vector Machine (SVM), Logistic Regression (LR), and multi-layer perceptron are used to identify personality types and prove to be discriminative for personality prediction. Twitter dataset is a large corpus of 1.2M tweets of 1.5K users (Plank and Hovy, 2015). Experiments performed by the dataset creators show that linguistic features are reliable representatives for two out of four personality dimensions. We hypothesize that the cause of the discrepancy is the difference between the distribution of personality types in social media users and the general U.S. population. Finally, Kaggle dataset collects the user posts of the Personality Cafe\textsuperscript{1} forum and covers 8.6K different people with 16 MBTI personality types (Mitchell, 2017).

3 Dataset

We use Reddit9k, Twitter, and Kaggle Myers-Briggs personality type datasets to train and evaluate our proposed model for automatic personality type prediction. In all datasets, the annotation process relies on self-reported personality types, and no questionnaire is given to the users. Previously, MyPersonality created from Facebook user data was a questionnaire-based dataset. However, it is not available to the public anymore. We make a unified dataset from the collection of the three available MBTI personality datasets and remove

\textsuperscript{1}https://www.personalitycafe.com
the non-English contents. We find that the new dataset is highly skewed towards two out of four personality dimensions. There are a few reasons for that. i) According to Plank and Hovy (2015) the distribution of personality types among the United States population is not balanced. ii) Users from some specific personality types tend to participate in social media platforms and express their personality types more than others. Our experiments also show that the class imbalance highly affects training, generating poor results for small classes, among evaluation methods. To alleviate the skewness of the data in training we take two (standard) steps: add class weights concerning their size in loss computation and make a balanced subset of the original dataset. We notice that the former does not improve the performance significantly, but the latter does. Hence, we create a balanced version of the dataset by over-sampling the small and under-sampling the large classes such that their final sizes become equal to the original average size of the 16 MBTI personality types before sampling. Table 1 reports the unified personality dataset statistics after balancing.

### Table 1: Unified personality dataset statistics; p.: personality; # of dimensions: 4; # of types: 16

<table>
<thead>
<tr>
<th>Set</th>
<th>Size</th>
<th>Size/p.type</th>
<th>Size/p.dim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>558,352</td>
<td>34,897</td>
<td>279,176</td>
</tr>
<tr>
<td>Dev</td>
<td>79,776</td>
<td>4,986</td>
<td>39,888</td>
</tr>
<tr>
<td>Test</td>
<td>159,520</td>
<td>9,970</td>
<td>79,760</td>
</tr>
</tbody>
</table>

4 Personality Embedding

We build a general model to predict four MBTI personality dimensions and to infer personality embedding. The MBTI dimensions are expressed as Booleans (0/1). The personality dimensions are IE, or Introversion (I)/Extroversion (E); NS, or iNtuition (N)/Sensing (S); FT, or Feeling (F)/Thinking (T); and JP, or Judging (J)/Perceiving (P). Under this scheme, each instance can have multiple labels with four classes. The combination of these four classes gives $2^4 = 16$ MBTI personality types. We consider multi-labeling classification to learn the personality dimensions together. Our experiments show that sub-sampling creates a small training set with poor final results while over-sampling creates a huge dataset with hundreds of redundant examples. So, the models cannot differentiate the 16 classes (personality types) with the under-sampled small training data or they fail to predict the unseen examples of minority classes correctly in the over-sampled dataset as they get over-fitted by the redundant examples. Finally, we find that by converting the 16 classes into four in the multi-labeling scheme and applying over-sampling and sub-sampling simultaneously we can better overcome the class imbalance. The following section describes the proposed personality prediction model.

4.1 PersBERT Model

The use of pretrained language models and transformers shows significant improvements in various NLP problems. The bidirectional based transformer models such as BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019) overcome previously published language models trained on one direction (e.g. ULMFiT) (Howard and Ruder, 2018) or the shallow concatenation of left and right direction of text input (e.g. ELMo) (Peters et al., 2018b) in various text classification tasks. We use BERT architecture as the basis of our personality prediction model. BERT takes position, segment, and token embedding as input to compute the importance of a token in a sequence. For personality classification purposes, we take into account the sentiment of sentences in an input sequence aside from the standard BERT input. According to (Tausczik and Pennebaker, 2010), the level of emotion and sentiment expression by people in their opinions and the way they express their emotions define how people feel about the world. People’s everyday language is a rich source of their beliefs, thinking patterns, and personality. Because personality speaks of stable differences in characteristic patterns of thinking, feeling, and behaving, it is connected with emotion and sentiment (Corr and Matthews, 2009). In this regard, some tools such as LIWC are designed to organize the words in psychologically meaningful categories and to identify emotion in language. They are also widely used in Psycholinguistic studies (Tausczik and Pennebaker, 2010). The connec-
tion between language, emotion, and personality elevates opinionated user generated content into a valuable resource for mining people’s personalities.

In our approach we split the input sequence into linguistic sentences.\(^2\) The sentiment of each sentence is one of positive, negative, or neutral; it can be inferred using any sentiment analysis tool. We give the utilized tools in Section 4.3. The sentiments of input tokens are embedded using a \(3 \times k\) matrix that is randomly initialized, where \(k\) is the size of the hidden states of the model. Then, these sentence-wise sentiment embeddings are accumulated with the three standard embeddings of the BERT model \((E_t^\text{token}, E_t^\text{position}, E_t^\text{segment})\) to form the input embedding \((E_t\) for token \(t)\). So, \(E_t = E_t^\text{token} + E_t^\text{position} + E_t^\text{segment} + E_t^\text{sentiment}\).

Figure 1 shows the model architecture. The input embeddings are given to the BERT sentence classification model that takes a sequence of linguistic sentences as one single input compared to the sentence-pair model that takes two inputs (e.g. a question and its answer). A fully connected layer forms a classifier that squeezes the pooled output \((x)\) of the BERT model to four personality dimensions \((I/E, N/S, F/T,\) and \(J/P)\). The hidden state of [CLS] token \((h_{\text{[cls]}})\) is used as the input of the pooling layer. So, \(x = \tanh(W_p h_{\text{[cls]}} + b_p), \logit = W^c x + b^c\) where \(W^c, W^p, b^p,\) and \(b^c\) are the layers’ parameters. Similar to other multi-label multi-class problems, the loss is the overall binary cross entropy among all classes, \(L = \frac{1}{N} \sum_{i \in N,c \in C} y_{c,i} \log \sigma(y_{c,i}^\prime) + (1 - y_{c,i}) \log (1 - \sigma(y_{c,i}^\prime))\).

where \(N\) is the number of examples, \(C\) number of classes, \(\sigma\) sigmoid function and \(y, y^\prime\) are true labels and logits (input of probability function) respectively. We refer to the proposed model as PersBERT for the rest of the paper.

4.2 Multi-class Multi-label Baselines

We mentioned earlier that personality is connected to emotion and sentiment (Tausczik and Pennebaker, 2010; Corr and Matthews, 2009). Also, automatic prediction of MBTI personality is being considered under a multi-label setting. Thus, we choose baselines with various architectures that are widely used in sentiment analysis or multi-label classification. They are listed as follows: Kim-CNN & XML-CNN are two CNN-based neural network models. The former is one of the initial and successful applications of Convolutional Neural Network (CNN) for text classification (Kim, 2014). And the latter is designed for extreme multi-label text classification where the number of labels can exceed even a few thousand (Liu et al., 2017). Its architecture inherits Kim-CNN’s model specification with an additional dynamic max-pooling layer that highlights important information across different parts of a document. XML-CNN was able to beat most of the deep learning baselines in six benchmark datasets. DocBERT is the BERT model with a fully connected layer that converts the hidden state of the BERT pooling layer to \(C\) activations for \(C\)-class classification (Devlin et al., 2019). The pooling layer pools the model by taking the hidden state corresponding to the classification token ([CLS]) of the input sequence through non-linearity (tanh). We fine-tune DocBERT for classification and initialize it with pretrained BERT-base-uncased weights. Lastly, Hierarchical Attention Network (HAN) is a recurrent neural network model that mirrors the hierarchical structure of the English language (Yang et al., 2016). Applying attention mechanisms in word and sentence-level enables this model to find crucial parts of the document for the downstream classification task. The model outperforms its competitive baselines in sentiment analysis of user reviews dataset including Yelp, Amazon, and IMDB.

4.3 Evaluation

We train the models on 30 epochs with the batch size of 16 or 32. Training is controlled by early stopping with patience = 5, which will be stopped after 5 consequent epochs of no improvement of the highest F1 score gained. The test set is evaluated using the model with the best F1 of the dev set. We use Google News (GNews) (He et al., 2020; Liu et al., 2015) and FastText token embedding in our experiments for the two CNN-based (Kim-CNN & XML-CNN) and LSTM-based baselines (HAN) (Mikolov et al., 2018). However, DocBERT and PersBERT models’ parameters are initialized with their corresponding BERT-base-uncased model weights. The BERTAdam optimizes these two models with the learning rate of \(2e - 5\) recommended in (Devlin et al., 2019). We set the sequence length = 256 for all models. Similar to DocBERT, all parameters of PersBERT are updated during backpropagation. Training PersBERT with more than \(5K\) examples (Table 1) takes

\(^2\)we use NLTK sentence tokenizer (Loper and Bird, 2002).
Table 2: Personality prediction on the unified dataset of Table 1; Jacc.:Jaccard, Hamm.:Hamming

<table>
<thead>
<tr>
<th>Method</th>
<th>Jacc.</th>
<th>Hamm.</th>
<th>Ma.-F1</th>
<th>Mi.-F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kim-CNN, GNews</td>
<td>46.82</td>
<td>41.76</td>
<td>62.82</td>
<td>63.78</td>
</tr>
<tr>
<td>Kim-CNN, FastText</td>
<td>45.83</td>
<td>39.31</td>
<td>62.23</td>
<td>62.86</td>
</tr>
<tr>
<td>XML-CNN, GNews</td>
<td>44.72</td>
<td>45.76</td>
<td>56.69</td>
<td>61.80</td>
</tr>
<tr>
<td>XML-CNN, FastText</td>
<td>47.97</td>
<td>40.96</td>
<td>64.0</td>
<td>64.83</td>
</tr>
<tr>
<td>HAN, GNews</td>
<td>46.62</td>
<td>41.18</td>
<td>63.03</td>
<td>63.59</td>
</tr>
<tr>
<td>HAN, FastText</td>
<td>46.29</td>
<td>38.48</td>
<td>62.83</td>
<td>63.29</td>
</tr>
<tr>
<td>DocBERT</td>
<td>86.03</td>
<td>7.46</td>
<td>92.47</td>
<td>92.49</td>
</tr>
<tr>
<td>PersBERT</td>
<td>86.97</td>
<td>6.94</td>
<td>93.03</td>
<td>93.03</td>
</tr>
</tbody>
</table>

5 days on a TITAN RTX GPU with batch size=16.

For evaluating multi-label personality prediction, we use Jaccard Similarity, Hamming loss, Macro-F1, and Micro-F1 scores. For more information about the measures the reader is directed to (Wu and Zhou, 2017). We use scikit-learn library (Pedregosa et al., 2011) for evaluation measures and other statistical methods. We utilize VADER, a rule-based model for the general sentiment analysis task, to infer the sentiment of sentences (Hutto and Gilbert, 2014). VADER gives us a compound sentiment score between -1 and +1. The scores between -1 and -0.05 indicate negative, the ones greater than 0.05 show positive sentiment, and the scores between -0.05 and +0.05 have a neutral sentiment. Each token inherits the sentiment of the sentence in which the token appears. For the two classification ([CLS]) and separator [SEP] tokens, we use neutral embedding. Although VADER is a token-based sentiment tool, we use sentence-wise sentiment instead of token-wise for two reasons: i) our intuition is to let the model learn the transition of sentiment across sentences and not tokens; this follows from the assumption that the change of sentiment from sentence to sentence may indicate one’s personality. ii) BERT uses sub-words units known as Word-pieces and each VADER lexicon may be composed of multiple Word-pieces. Thus, we must assign the sentiment of an entry in VADER lexicon to all its Word-pieces. For example, the sentiment of ‘huggable’ must be assigned to its three sub-words in our model: ['hug', '##ga', '#ble']. Also, our experiments on the dev set show that token-wise sentiment avoids learning the transition of sentiments and does not improve the model performance as much as sentence-wise sentiment.

Experimental results of multi-label MBTI personality prediction on the unified dataset (Table 1) are provided in Table 2. They indicate that PersBERT trained on 256 tokens of input sequence achieves the best results among baselines in all multi-class-multi-label evaluation measures. An F1 improvement of +0.5% and −0.52% Hamming loss reduction on ≥ 159K test instances compared to DocBERT shows that adding sentiment embedding of sentences to the input distinguishes the personality types more accurately. Apart from that, both transformer-based models, DocBERT and PersBERT, show significant improvement over the two CNN models with two different pretrained embeddings as well as the HAN (about +40% in Jacc. and −30% in Hamming loss). We believe that the two masking and next sentence prediction techniques used in BERT’s pretraining enable the model to better understand the relationship between the words of a sentence in both directions, as well as the relationship between the sentences. This leads to an enriched language model and a remarkable improvement in identifying personality signals from individuals’ language compared to other baselines.

5 Transfer Learning

We aim to study if the knowledge gained from our personality prediction model, PersBERT, helps opinion-oriented problems. Personality is closely connected with opinions and how people form opinions; hence, we choose three tasks, i.e., hyperpartisan news detection, authorship verification, and stance detection (Hosseinia et al., 2020), that are designed around opinion mining. We create our transfer learner named DocBERT + PersBERT by connecting the pretrained personality model, PersBERT, to DocBERT (Figure 2). Empirical results show that these transformer models achieve the best results among the baselines introduced in Section 4.2. We share a fully connected layer, classifier layer, between the DocBERT and PersBERT models. The layer takes the concatenation of the
Table 3: Hyperpartisan news dataset

<table>
<thead>
<tr>
<th>News dataset</th>
<th>train</th>
<th>dev</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>by-article</td>
<td>386</td>
<td>129</td>
<td>130</td>
</tr>
<tr>
<td>by-publisher</td>
<td>164,944</td>
<td>118,510</td>
<td>149,794</td>
</tr>
</tbody>
</table>

output vector of pooling layers and converts them to $C$ classes. Recall that in DocBERT and PersBERT, the pooling layer pools the model by taking the hidden state corresponding to [CLS] token of input sequence using a non-linear activation (tanh). Hence,

$$a_{\text{PersBERT/DocBERT}}^{\text{[cls]}} = \tanh(W_{p/d}^{\text{[cls]}} + b_{p/d}),$$

$$z = W_{c}[x_{\text{DocBERT}}; x_{\text{PersBERT}}] + b_c$$

where $W_{p/d}, b_{p/d}$ are the parameters of PersBERT or DocBERT pooling layers. $[; ;]$ denotes concatenation, $W_{c}$ is the $2 \times k \times C$ classifier weight matrix and $k$ is the size of the pooled vectors (hidden state). Finally, the classifier output, $z$, is normalized with a Softmax function for downstream $C$-class classification tasks. Next, we introduce the three text classification problems for our evaluations.

5.1 Hyperpartisan News Detection

The term “hyperpartisan news” is used to define the extremely biased news in favor of the right or the left political spectrum. SemEval2019 task 4 proposes hyperpartisan news detection and has released only the training and dev sets of two versions of the hyperpartisan news dataset. In the first version, “news by-publisher”, all articles are labeled by the overall bias of the publisher as provided by BuzzFeed or MediaBiasFactCheck.com while in “news by-article” dataset documents are labeled manually by the agreement of the journalists (Kiesel et al., 2019). Because the test set is not released yet, we use SemEval dev set as test and split its training set into new training and dev sets with no publishers in common. Likewise, we create new sets for the “news by-article” dataset. Our topic modeling analysis on the “news by-publisher” training set reveals that it is highly imbalanced in terms of news classes. We use Non-negative Matrix Factorization (NMF) to estimate topic distribution in news. For some topics, top documents belong to only one class. To avoid the models to learn topics but hyperpartisanship we sample from the training set so that the resulting set includes an equal number of unique examples per topic. We only apply sub-sampling on the training set and keep the dev and test set intact. This process increases the F1 score of DocBERT model by 5% on the dev set. Table 3 provides the dataset statistics.

5.2 Stance Detection

Stance detection identifies if an opinion supports an idea or contradicts it. We use the new version of Procon dataset (Hosseinia et al., 2019) in our evaluation. The dataset covers the argumentative opinions of different controversial issues, ranging from education and immigration to birth control. The dataset has 4,264 instances and we split it into (70%, 10%, 20%) for training, dev and test, respectively. As each instance in Procon dataset is a pair of a question about an issue and an opinion about it, we use the BERT sentence-pair model for both DocBERT and DocBERT + PersBERT models. Thus, the input of the two BERT-based models is formed as [CLS] question [SEP] opinion [SEP] where [CLS], [SEP] are reserved tokens used by BERT for classification and separation of the two input parts respectively (Devlin et al., 2019).

5.3 Authorship Verification

Authorship Verification (AV) identifies whether a pair of documents are written by the same author. It has applications in plagiarism detection, forensic analysis, and sockpuppet detection, to name a few. We examine our model on three standard PAN AV datasets. Each dataset contains one training and one test set. We split the original training set into (70%, 30%) for training and dev, and evaluate on the original test set. Similarly, each instance in the AV dataset is a pair of documents, so, we use the BERT sentence-pair model for both DocBERT and DocBERT + PersBERT models. The input is formed as [CLS] first document [SEP] second document [SEP] where documents are written by one or two unknown author(s) and may contain several (linguistic) sentences.

5.4 Results and Analysis

The settings, training strategy, and baselines are the same as PersBERT’s (Table2, Section4.3) for the three aforementioned opinion-oriented clas-
Table 4: Effect of personality in three opinion-oriented tasks; P: Precision; R: Recall

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DocBERT</th>
<th>DocBERT+PersBERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>R</td>
<td>F1</td>
</tr>
<tr>
<td>Procon</td>
<td>72.89</td>
<td>77.65</td>
</tr>
<tr>
<td>PAN2014E</td>
<td>65.09</td>
<td>69.70</td>
</tr>
<tr>
<td>PAN2014N</td>
<td>60.33</td>
<td>73.74</td>
</tr>
<tr>
<td>PAN2015</td>
<td>44.26</td>
<td>75.60</td>
</tr>
<tr>
<td>News by-art.</td>
<td>72.77</td>
<td>72.34</td>
</tr>
<tr>
<td>News by-pub.</td>
<td>61.48</td>
<td>38.33</td>
</tr>
</tbody>
</table>

Table 5: Effect of topic and personality based sub-sampling of training set on “news by-publisher”; B.Acc.: balanced accuracy; test and dev sets are the original sets

<table>
<thead>
<tr>
<th>Training set</th>
<th>Entropy Model</th>
<th>B.Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>News by-pub.</td>
<td>0.99</td>
<td>DocB.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DocB.+PersB.</td>
</tr>
<tr>
<td>topic “ecom”</td>
<td>0.2705</td>
<td>50.02</td>
</tr>
<tr>
<td>topic “life”</td>
<td>0.6837</td>
<td>52.82</td>
</tr>
</tbody>
</table>

Table 6: “News by-publisher” results with entropy-based sampling; *:p-value of McNemar’s test \(\leq 10^{-5}\)

<table>
<thead>
<tr>
<th>Train data</th>
<th>Entropy</th>
<th>P</th>
<th>R</th>
<th>F1</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>0.99</td>
<td>61.48</td>
<td>38.33</td>
<td>47.22</td>
<td>57.04</td>
<td>33.29</td>
<td>42.05</td>
</tr>
<tr>
<td>sub-sampled</td>
<td>0.09</td>
<td>61.69</td>
<td>44.09</td>
<td>51.43</td>
<td>58.03</td>
<td>47.16</td>
<td>52.04*</td>
</tr>
</tbody>
</table>

The experiments reported above show that the proposed approach is not useful in identifying hyperpartisanship in the “news by-publisher” dataset. However, we anticipate there are some connections between personality and hyperpartisanship as opinion forms a bridge between these two concepts. We design the following experiments to investigate the hidden connection. Articles of the large “news by-publisher” dataset (Table 3) cover a wide variety of topics. So, we investigate whether personality types vary in mainstream and hyperpartisan classes for separate news topics. We first model topics of the news training set using the Non-negative Matrix Factorization (NMF) algorithm for 20 topics. Then, we choose distinct articles for each topic and induce MBTI personality dimensions using PersBERT (Section 4.1). We select these two topics to minimize the influence of per-topic-personality distribution. Later in this section, we measure the relationship between personality distribution and the PersBERT model is trained on social media data while news data is formal and usually follows its publisher’s strict writing regulation. It may lead to hiding the author’s informal writing and personality features. This difference between the language of news and social media data challenges the effect of transfer learning between the two domains. Secondly, it is expected that personality distribution differs between mainstream and hyperpartisan classes for different news topics, similar to what we observed in stance detection. The following section provides additional experiments for hyperpartisanship detection.

5.4.1 A Deeper Look into Hyperpartisanship

Despite the improvements of F1 in “news by-article” results using personality information, we do not see the same effect on hyperpartisan “news by-publisher” results. There is a reduction by \(-5.17\%\) of F1 in “news by-publisher” when personality information is added. We hypothesize that there are two main reasons for this behavior: first,
news topics using entropy. As noted before, a tuple of four dimensions gives us an MBTI personality type. We plot 16 MBTI personality types versus two news classes for the two selected topics (Figure 4). According to the figure, there is a remarkable difference between several personality types of the two news classes for the topic “econ.” 100% of personality types 7 and 10 are from hyperpartisan and mainstream news classes, respectively; about 80% of personality type 4 belongs to the mainstream, while 70% of type 8 forms hyperpartisan news. However, the plot of topic “life” shows less difference in news distribution among personality classes. We report average entropy of the two news classes across all personality types to measure the difference of personality distributions between the two news classes (Table 5).

\[
\text{entropy} = \frac{1}{|T|} \sum_{t \in T} \sum_{i \in I} -p_{i,t} \log_2(p_{i,t})
\]

Where \( T \) is the set of all personality types, \( I = \{\text{Hyp., Main.}\} \), \( |\cdot| \) denotes the size, and \( p_{i,t} = \frac{n_{i,t}}{\sum_{i \in I} n_{i,t}} \) is the proportion of news class if in personality class \( t \). The smaller the entropy, the more the two news classes have different distributions among the 16 personality types. We train both DocBERT and DocBERT+PersBERT on about 500 articles from topic “life” and “econ,” separately and evaluate it on the original test set. According to Table 5, training on topic “econ” with lower entropy results in higher balanced accuracy of DocBERT+PersBERT with an improvement of > 5% for the sequence length = 256. It shows that topic-based sub-sampling gives us a more distinctive representation of personality types that contributes to better hyperpartisan news detection. On the other hand, training on the data with higher entropy (topic “life”) results in lower accuracy of DocBERT+PersBERT compared to DocBERT indicating that adding unbiased personality features makes the differentiation between the two news classes harder. Moreover, we sub-sampled the whole training data such that the average entropy does not exceed the low amount of 0.1. The sampling gives us 111,614 training examples for \( \text{entropy} \leq 0.1 \). Results in Table 6 reveal that training on data with unbalanced personality distribution of news classes results in remarkable improvements for both models with higher F1 of DocBERT+PersBERT.

6 Conclusion

We believe that this work lays the foundation for leveraging personality signals in a variety of opinion-oriented tasks. We first proposed a novel model, PersBERT, that jointly models the sentence-specific sentiment and personality information building upon the BERT architecture to predict the MBTI personality dimensions. Our pretrained personality transformer improves BERT results and other baselines in benchmark datasets on the personality task. Further, our proposed model was
used on different downstream NLP tasks providing major improvements showing that the subtle signals of user sentiment and their connection with personalities captured by our model are useful in real-world NLP tasks. It is worthwhile to note that the performance comes from training using only short sequences of online user posts (i.e. noisy data for personality). We believe the improvements of our model can be more pronounced if trained upon large-scale gold standard personality datasets (e.g. curated using controlled experiments which is a potential future work). We find that personality signals are more distinctive in authorship verification and stance detection than hyperpartisan news detection where the data is sourced from formal and more supervised writings. However, our personality embeddings can still be used for an effective sub-sampling even in hyperpartisan news detection. Our architecture allows for novel analysis and insights that were previously unknown and have the potential to improve various other NLP tasks which we defer for future exploration.

Acknowledgments

Research was supported in part by grants NSF 1838147, NSF 1838145, ARO W911NF-20-1-0254. The views and conclusions contained in this document are those of the authors and not of the sponsors. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References


Qingyuan Liu, Eduard C. Dragut, Arjun Mukherjee, and Weiyi Meng. 2015. Florin: A system to support (near) real-time applications on user generated content on daily news. 8(12).


Barbara Plank and Dirk Hovy. 2015. Personality traits on twitter—or—how to get 1,500 personality tests in a week. In Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pages 92–98.


Application of Deep Learning Methods to SNOMED CT Encoding of Clinical Texts: From Data Collection to Extreme Multi-Label Text-Based Classification

Anton Hristov\textsuperscript{1,2}, Aleksandar Tahchiev\textsuperscript{1}, Hristo Papazov\textsuperscript{3}, Nikola Tulechki\textsuperscript{1}, Todor Primov\textsuperscript{1} and Svetla Boytcheva\textsuperscript{1,4}

\textsuperscript{1}Sirma AI (Ontotext), Bulgaria
anton.hristov@ontotext.com, alexander.tahchiev@ontotext.com, nikola.tulechki@ontotext.com, toodor.primov@ontotext.com, svetla.boytcheva@ontotext.com
\textsuperscript{2}Faculty of Mathematics and Informatics, Sofia University ”St. Kliment Ohridski”, Bulgaria
h.g.papazov@gmail.com
\textsuperscript{3}MIT, Mathematics PhD Student, USA
\textsuperscript{4}Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Bulgaria

Abstract

Concept normalization of clinical texts to standard medical classifications and ontologies is a task with high importance for healthcare and medical research. We attempt to solve this problem through automatic SNOMED CT encoding, where SNOMED CT is one of the most widely used and comprehensive clinical term ontologies. Applying basic Deep Learning models, however, leads to undesirable results due to the unbalanced nature of the data and the extreme number of classes. We propose a classification procedure that features a multiple-step workflow consisting of label clustering, multi-cluster classification, and clusters-to-labels mapping. For multi-cluster classification, BioBERT is fine-tuned over our custom dataset. The clusters-to-labels mapping is carried out by a one-vs-all classifier (SVC) applied to every single cluster. We also present the steps for automatic dataset generation of textual descriptions annotated with SNOMED CT codes based on public data and linked open data. In order to cope with the problem that our dataset is highly unbalanced, some data augmentation methods are applied. The results from the conducted experiments show high accuracy and reliability of our approach for prediction of SNOMED CT codes relevant to a clinical text.

1 Introduction

The task of automatic encoding of clinical text with standard medical classifications and ontologies is with high importance for healthcare organizations and medical research. Truly, more than 80\% of clinical documents are stored in free-text format. This paper presents a research effort in solving the problem of automatic encoding of textual description of medical diagnoses with one of the most widely used (Lee et al., 2013) and comprehensive ontologies – the Systematized Nomenclature of Medicine – Clinical Terms (SNOMED CT)\textsuperscript{1}. One of the most important characteristics of SNOMED CT, which makes it significantly different from the rest of the standard medical classifications, is that it is based on compositional grammar\textsuperscript{2}. Another aspect of popularity and importance of SNOMED CT for health information is interoperability, that is discussed in (Peterson and Liu, 2020). SNOMED CT is well known for being one of the most comprehensive medical ontologies, which makes the task of automatic encoding an extreme scale classification task with more than 360,000 medical codes. Currently, this task has not been solved with sufficient accuracy for all possible classes. Usually the developed solutions cover restricted terminology from 10 to a couple of thousands terms (Gaudet-Blavignac et al., 2021). The compositional nature of the SNOMED CT codes gives us the opportunity to address the problem either with classical approaches for classification tasks or with specific solutions that benefit from the compositional grammar’s structure. As a product of our research,

\textsuperscript{1}https://www.snomed.org/
\textsuperscript{2}https://confluence.ihtsdotools.org/display/DOCS/Compositional+Grammar+-+ Specification+and+Guide
the developed service for automatic encoding with SNOMED CT codes will be used mainly for Electronic Health Records (EHR) processing for patients with oncological diseases and certain rare diseases. Most of these diseases are well known to have a huge number of related diseases. Thus, our study will not be restricted only to the diseases of interest but will have a much broader scope. We propose an adaptation of the approach proposed by (Chang et al., 2020) and demonstrate the entire process from training dataset construction to classification model design and training.

2 Related Work

The problem of automatic encoding of EHR with SNOMED CT codes was investigated by many researchers since the very beginning of the ontology development. Different solutions cover broad range of SNOMED CT codes from 10 to a couple of thousands, usually the main obstacle for scalability is the availability of sufficient volume of annotated training data. Basaldella et al (Basaldella et al., 2020) present COMETA - manually annotated corpora by experts that contain 20k English biomedical entities encoded with SNOMED CT.

The most popular approaches for automatic encoding, include hybrid methods combining regular expressions and vector space models (Ruch et al., 2008) with top precision 0.823 and mean avg precision 0.45 for 1239 MEDLINE citations. Some approaches take in consideration compositional structure (Liu et al., 2012) of SNOMED CT.

Recent research is based on deep learning techniques, and the most promising solutions are using transformers like BERT (Devlin et al., 2018). Pattisapu et al (Pattisapu et al., 2020) apply word embeddings, graph embeddings and BERT derivatives transformers and achieve the highest accuracy 0.83 for two benchmark datasets CADEC and PsyTAR.

Kraljevic et al (Kraljevic et al., 2021) propose MedCAT with Macro F1: 0.841–0.860 across different clinical domains and tasks. MedCAT is based on Word2Vec embeddings, and there is also MedCAT BERT version based on clinicalBERT (Alsentzer et al., 2019), and latter model shows a little bit lower performance than the former one.

A recent systematic review (Gaudet-Blavignac et al., 2021) shows that only few of the developed services for automatic encoding with SNOMED CT, are provided as open source - The clinical Text Analysis and Knowledge Extraction System (cTAKES) (Savova et al., 2010) and MetaMap (Aronson, 2001). Both of them are rule-based.

3 Data

One of the key factors that plays a role in the automatic encoding of SNOMED CT codes is the data. In our project, we do not have annotated data which can be used to train the developed models. Thus, we use certain public data and linked open data in order to automatically generate annotated corpora that can serve as a training dataset.

3.1 Data Sources

In our research, we will consider only a subset of the available SNOMED CT codes, namely those related to disorders, clinical findings and procedures. The relevant medical ontologies, standard classifications, and vocabularies for the project, which are used to enrich the SNOMED CT descriptions with additional alternative textual descriptions, are the Human Disease Ontology3, the International Classification of Diseases, 10th revision (ICD-10)4, the International Classification of Diseases, 9th revision (ICD-9)5, the International Classification of Diseases for Oncology, 3rd Edition (ICD-O-3)6, the Medical Subject Heading (MESH)7, the Mondo Disease Ontology (MONDO)8, the Orphanet Rare Disease Ontology (ORDO)9, and the Unified Medical Language System (UMLS)10. Benefiting from the resources provided by the linked open data cloud (LOD)11, we can identify some of the mappings between the ontologies listed above using Bioportal12. Some general equivalence mappings are provided for ICD-10 and ICD-913 as well as rules for mappings between SNOMED CT and ICD-1014. In addition, the ICD-10 CM Alphabeti-
cal Index\textsuperscript{15} is used. All these resource and official mappings between them provide valid encoding of the textual descriptions of diseases with SNOMED CT codes.

3.2 Data Integration

As we alluded above, in order to increase the size of the corpus of SNOMED codes with textual descriptions (and hence boost the predictive power of our neural network by ensuring a richer training set), we extract mappings between SNOMED CT and other medical ontologies, standard classifications, and vocabularies. Then, we use these mappings to link SNOMED CT codes to descriptions native to the aforementioned resources.

Following this guiding principle, but applying it to different subsets of SNOMED CT codes and allowing different degrees of description transitivity (stay tuned), we constructed four distinct datasets, the last two of which allowed for a high classification accuracy.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Corpus Size</th>
<th>SNOMED CT Codes</th>
<th>Unique Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset V1</td>
<td>22M</td>
<td>128k</td>
<td>280k</td>
</tr>
<tr>
<td>Dataset V2:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Procedures</td>
<td>626k</td>
<td>227k</td>
<td>469k</td>
</tr>
<tr>
<td>Findings</td>
<td>106k</td>
<td>64k</td>
<td>105k</td>
</tr>
<tr>
<td>Disorders</td>
<td>140k</td>
<td>65k</td>
<td>107k</td>
</tr>
<tr>
<td>Dataset V3</td>
<td>85k</td>
<td>14k</td>
<td>54k</td>
</tr>
<tr>
<td>Dataset V4</td>
<td>198k</td>
<td>14k</td>
<td>58k</td>
</tr>
</tbody>
</table>

Table 1: Dataset Evolution

We can conceptualize the overarching principle behind the construction of the different datasets as follows: First, we choose a certain subset of SNOMED CT codes whose elements will serve as labels in the classification procedure. Second, we consider the medical codes from the above ontologies, classifications, and vocabularies that are linked to our chosen SNOMED CT subset through an "exact-match" type predicate.\textsuperscript{16} Third, we build a graph whose vertices are all of the chosen SNOMED CT codes and their "exact-match" neighbors; and whose edges are precisely these "exact-match" mappings. Fourth, we prescribe a degree of description transitivity. That is, we specify whether medical codes in connected components will share all textual descriptions associated with that component or simply the descriptions associated with their immediate neighbors. Finally, we extract a corpus of SNOMED CT codes along with the natural language descriptions that these codes acquired from the mapping graph.

Version 1 of our dataset reflected the naïve idea of considering a graph with full description transitivity. Of course, this approach of total transitive search between ontologies is largely misguided since the mappings between SNOMED CT codes and outside resources are rarely one-to-one. Indeed, these mappings prescribes similarity rather than identity – a circumstance that caused the graph generated by the V1 SNOMED CT subset to contain a connected component encompassing more than 90\% of the vertices. Thus, the majority of the relevant SNOMED CT codes became indistinguishable for our classification procedure.

For the second version of our dataset, we considered an even larger subset of SNOMED CT codes, which we further split into three categories – Procedures, Findings, and Disorders – respecting the official SNOMED CT organization. We used these three subsets to generate three distinct mapping graphs following the above procedure. Again, we prescribed full mapping transitivity on the individual graphs, but we forbade communication between graphs. This new strategy decreased the edge density of the graphs considerably, but large connected components were still present, which led to poor classification performance.

Consequently, we decided to narrow down the generating subset of SNOMED CT codes and to...
allow description-sharing only among immediate "exact-match" neighbors. Thus, for Dataset V3, we considered solely the SNOMED CT codes which exactly matched the following widely encountered procedures, findings, and disorders related to oncological diseases, certain rare diseases, and digestive, neurological, and respiratory diseases.

The lack of description transitivity in V3 caused the majority of the considered SNOMED CT codes to have unique description clusters. This circumstance allowed us to observe high classification accuracy for the first time. However, many SNOMED CT codes were now matched to a single single-word textual description, and the augmentation strategies discussed in the next subsection failed to meaningfully increase the description clusters of such codes.

For that reason, the official verified SNOMED CT ontology mappings used in V3 were supplemented with additional mappings excavated from Wikidata\(^\text{17}\). Then, full description transitivity was applied, and thus Dataset V4 came to be.

<table>
<thead>
<tr>
<th>Additional Data</th>
<th>Corpus Size</th>
<th>SNOMED CT Codes</th>
<th>Unique Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>112k</td>
<td>8k</td>
<td>3k</td>
</tr>
</tbody>
</table>

Table 2: Additional Data

### 3.3 Data Augmentation

Our data integration strategies resulted in a one-to-many mapping of SNOMED CT codes to synonymous natural language descriptions. This mapping, however, featured a significant number of SNOMED CT codes with less than four textual descriptions. In order to address this circumstance, which would have otherwise interfered with the precision of our neural network, we employed several data augmentation techniques aimed at synthetically increasing the set of descriptions so that each SNOMED CT code could get mapped to at least four descriptions.

Our augmentation strategies were motivated by considerations of what synonymous textual descriptions could arise in the work of medical professionals.

#### 3.3.1 Random Swap and Random Synonym Insertion

We adapted some of the code developed for Easy Data Augmentation\(^\text{(Wei and Zou, 2019)}\) for the purposes of random word swapping and random synonym insertion. The Random Swap transformation works by selecting two random indices in a list of multiple words, and then, swapping the words with the corresponding indices. Only one swap is performed per transformation, which guarantees that novel descriptions are produced after the augmentation. The Random Synonym Insertion transformation works by shuffling the words in a sentence and looping over the shuffled sequence of words until a word with a WordNet\(^\text{18}\) synonym is selected. Once such a word is found, a random synonym is pulled from its list of synonyms and inserted at a random place in the initial sentence. If no synonyms are found, nothing happens.

**Examples:**

- **Random Swap:** Fear of thunderstorms → Fear thunderstorms of.
- **Random Synonym Insertion:** Complete loss of teeth due to trauma → Complete loss of hurt teeth due to trauma.

#### 3.3.2 Typographical Augmentation

Since medics work in tense environments, they are susceptible to making errors while typing medical records, as they are subjected to a lot of stress, strain, and lack of sleep. We have developed the following augmentations mimicking potential typos:

- **Swap adjacent character:** Syndrome → Synrdome.
- **Remove character:** Syndrome → Syndome.
- **Change character with corresponding adjacent keyboard-key character:** Syndrome → Syndrone.
- **Insert adjacent keyboard-key character to a word:** Syndrome → Syundrome.

All these augmentations are applied on a randomly selected character of a randomly selected word.
3.3.3 Manual Augmentation

At certain places, where the above strategies could not be naturally applied, synonymous natural language descriptions were manually crafted.

Example:

3-PGDH deficiency $\rightarrow$ 3-phosphoglycerate dehydrogenase deficiency.

4 Text-Based Encoding with SNOMED CT Codes

If we have an unbalanced dataset, or if we want to split our problem into sub-problems, we can group our labels into clusters and train a model to predict to which of the clusters each sample belongs. After that, another model can refine (map) every predicted cluster to a specific label.

The proposed approach of text-based SNOMED CT classification is the following (see Fig. 2):

- Data Augmentation
- Label Clustering
- Sampling
- Train Multi-Cluster Classification Model
- Train Model for Clusters to Labels Refinement

4.1 Data Augmentation

The data augmentation techniques used in this step are described in detail in the previous section. The following parameters are used:

$$\text{augment probability} = 40\%$$
$$\text{minimum samples} = 5,$$

where augment probability refers to what portion of the current description’s words will be augmented and minimum samples refers to the minimum number of description samples every class should contain after the augmentation.

4.2 Label Clustering

Label clustering is widely used in extreme scale classification problems because datasets are mainly unbalanced and there are vast numbers of classes or because we want the classification task to be performed with less granularity. Our approach for clustering the dataset labels into groups is done by label embeddings, used in (Khandagale et al., 2020), and by applying clustering algorithm to it. Label embeddings for specific labels can be produced by summing all sample embeddings for which it is active. So, if we denote $X$ to be the matrix holding the embeddings of all samples’ descriptions, $X = [\text{samples} \times \text{embeddings}]$, and $Y$ to be the matrix holding multi-hot encoding of samples’ classes, $Y = [\text{samples} \times \text{labels}]$, Label embeddings (matrix $L$) is calculated using dot product between $X$ and $Y$. This matrix $L$ gives us information on how each label relates to each sample in our data, $L = Y^T X = [\text{labels} \times \text{embeddings}]$. For encoding the input samples’ descriptions, we applied the pre-trained BioBERT model (Lee et al., 2020). Clustering is done by a K-Means algorithm, with selected number of clusters of 100. This specific number is selected by manual analysis of data distribution over different number of clusters. The desired number of clusters is the smallest number that produces the minimum number of labels contained in more than one cluster, as well as best distribution of the labels for each cluster.
4.3 Sampling
Since our dataset is highly unbalanced because of the specifics of the domain, there are classes with only 5 samples and there are classes with more than 800 samples. The naïve sampling of random data splitting to train/dev/test sets will not work because this will result in all samples of some classes to be included entirely into one of the splits, which will lead to reduced accuracy. So, we developed a custom sampling strategy, which extracts distributed number of samples by a random manner into the dev and test split, which will result specific classes included into the dev and test splits to be included into the training corpus. For example, if class \( N \) has only 5 samples, its samples will be distributed as follows: train/dev/test = 3/1/1.

4.4 Train Multi-Cluster Classification Model
After label clustering is applied on the data, the next step is training a model for classification of the produced clusters. This can be formulated as binary, multi-class or multi-label classification. Since our dataset is very complex, and one class can be included into one or multiple clusters, we modified the official BioBERT implementation to perform a multi-label classification using Area Under the Curve (ROC AUC) as a scoring function. As an input, we have transformed our dataset using the produced clusters instead of the original labels, and we have fine-tuned the BioBERT weights on it.

4.5 Train Model for Clusters to Labels Refinement
The goal of the last step in the pipeline is finding to which label every sample belongs based on the already predicted cluster. There are a lot of possible solutions for mapping the sample’s predicted clusters to their labels, and a classical one is for each label to look into all instances which is too expensive, discussed in (Chang et al., 2020). Another approach is training multiple models. A model for every label including the subset of all instances included into the predicted clusters (Chang et al., 2020), which will lead to the number of models equal to the number of classes. Since we are dealing with an extreme scale classification task with more than 10k classes, we think that this is not practical for applying it in real applications. We have trained linear one-vs-rest classifiers (Support Vector Classification (Platt, 1999), (Chang and Lin, 2011)) for every cluster including all its instances with BioBERT sample embeddings as an input. This approach results in 100 SVC trained models for clusters-to-labels refinement using Area Under the Curve (ROC AUC) as a scoring function. Our contribution:
- Proposed augmentation techniques matching data distribution specifics;
- Proposed sampling strategy dealing with unbalanced data;
- Multi-class cluster classification is replaced with multi-label cluster classification, increasing the task’s level of complexity;
- Cluster to label refinement is compressed to model per cluster, which is more suitable for extreme scale classification tasks;

5 Experiments and Results
5.1 Dataset V1
On version 1 of our dataset, we initially attempted a classical multi-class classification approach by using pretrained BioBERT (Lee et al., 2020). The results were close to random guessing, so we tried another approach. We used some standard community detection algorithms (like Louvain (Blondel et al., 2008), (Dugué and Perez, 2015), (Traag et al., 2011) and Leiden (Traag et al., 2019)) to group SNOMED CT codes into classes in order to train a cascade of hierarchical BioBERT classifiers. This grouping was necessary because 95% of Dataset V1 forms a dense graph (see Figure 5). After analyzing the results, we concluded that by this grouping a lot of important connections were removed. For this reason, we have left this approach aside.

5.2 Dataset V2
On the new version of the dataset, we tried to solve the problem by a multi-label classification approach using pretrained BioBERT again. After comprehensive training iterations, our model reached Area Under the Curve (ROC AUC) of 0.60 (Figure 6) which was not high enough for solving the problem.

5.3 Dataset V3
The proposed approach described in section 4 is applied on this third version of our dataset. We have fine-tuned the BioBERT weights for the multi-cluster classification task (Step 4 of our pipeline),
Figure 3: Cluster classification BioBERT model is dealing with multi-label data

Figure 4: Refinement maps every sample with its corresponding labels based on the sample’s predicted cluster

Figure 5: Dataset V1 - Dense Graph

and after 7 epochs of training, it reached Area Under the Curve (ROC AUC) of 0.99653. Training

Figure 6: Dataset V2 - ROC curve

SVC for every cluster (100 SVC models) produces Area Under the Curve (ROC AUC) of 0.83273.

5.4 Dataset V4

The proposed approach described in the previous section is applied on this fourth version of our dataset. The Vocabulary is based on BioBERT v1.1, trained over PubMed. Some of the characteristics of Dataset V4 are presented in Table 3.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique tokens</td>
<td>6,522</td>
</tr>
<tr>
<td>Number of tokens</td>
<td>2,949,353</td>
</tr>
<tr>
<td>Min of tokens</td>
<td>1</td>
</tr>
<tr>
<td>Max of tokens</td>
<td>189</td>
</tr>
<tr>
<td>Mean of tokens</td>
<td>14.85</td>
</tr>
<tr>
<td>Median of tokens</td>
<td>11.0</td>
</tr>
</tbody>
</table>

Table 3: Dataset V4 Characteristics

We have fine-tuned the BioBERT weights for the multi-cluster classification task (Step 4 of our pipeline), and after 25 epochs of training, it reached Area Under the Curve (ROC AUC) of 0.977.

Training SVC for every cluster (100 SVC mod-
els) produces Area Under the Curve (ROC AUC) of 0.804.

5.5 Discussion
The comparison (see Table 4) of the bioBERT multi-label classification with bioBERT clustering and label refinement, show that the proposed approach significantly improves the accuracy for SNOMED CT encoding task. The experiments are performed for trained models for 7 epochs for Dataset V2 - disorders subset.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>bioBERT multilabel classification</td>
<td>0.56</td>
</tr>
<tr>
<td>bioBERT clusterings + label refinement (final approach)</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Table 4: Comparison of the bioBERT models for Dataset V2- disorders

The proposed approach shows high accuracy and scalability. The additional steps for label refinement do not cause significant slow down of the learning process of the model. The obtained accuracy of the given method shows a significant improvement of the evaluation, compared to other solutions in the literature of the same problem, that report accuracy in the range from 0.83 (Pattisapu et al., 2020) up to 0.86 (Kraljevic et al., 2021). Moreover, the presented results of the experiments and the evaluation are for a larger dataset and a wider range of SNOMED CT codes.

6 Conclusion and Further Work

We demonstrated how can be generated annotated dataset with SNOMED CT codes. The proposed approach demonstrates high accuracy and scalability. In comparison with other state-of-the-art approaches the achieved accuracy for the proposed model is relatively high and more over for wider coverage of SNOMED CT.

Our further work includes training of Multilingual BERT to solve the multilingual problem. Possible increase of the Area Under the Curve (ROC AUC) scores can be achieved through grid search applied to the selection of the K-Means clusters number, until finding the optimal number, based on the distribution of the data.

Acknowledgments

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 825292 (ExaMode, http://www.examode.eu/).

References


Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A library for support vector machines. 2(3).


Syntax Matters! Syntax-Controlled in Text Style Transfer

Zhiqiang Hu  
University of Electronic Science and Technology of China  
hzq950419@gmail.com

Roy Ka-Wei Lee  
Singapore University of Technology and Design  
roy_lee@sutd.edu.sg

Charu C. Aggarwal  
IBM T. J. Watson Research Center  
charu@us.ibm.com

Abstract

Existing text style transfer (TST) methods rely on style classifiers to disentangle the text’s content and style attributes for text style transfer. While the style classifier plays a critical role in existing TST methods, there is no known investigation on its effect on the TST methods. In this paper, we conduct an empirical study on the limitations of the style classifiers used in existing TST methods. We demonstrate that the existing style classifiers cannot learn sentence syntax effectively and ultimately worsen existing TST models’ performance. To address this issue, we propose a novel Syntax-Aware Controllable Generation (SACG) model, which includes a syntax-aware style classifier that ensures learned style latent representations effectively capture the syntax information for TST. Through extensive experiments on two popular TST tasks, we show that our proposed method significantly outperforms the state-of-the-art methods. Our case studies have also demonstrated SACG’s ability to generate fluent target-style sentences that preserved the original content.

1 Introduction

Text Style Transfer (TST) is an increasingly popular natural language generation task that aims to change the stylistic properties (e.g., the sentiment of text) of the text while retaining its style-independent content (Hu et al., 2020). Due to the difficulty in obtaining training sentence pairs with the same content and differing styles, most existing methods are designed to perform TST in an unsupervised manner; the models only have access to non-parallel, but style-labelled sentences.

A popular TST approach is to leverage an adversarial learning autoencoder framework where a style classifier or discriminator is pre-trained to first disentangle the content and style latent representations, before using a decoder to generate the output sentence in the target style (Shen et al., 2017; Zhao et al., 2018; Fu et al., 2018; Chen et al., 2018). Another line of work proposed attribute-controlled generation methods where the style attribute latent vector is learned and combine with the latent representation of the text to generate output sentences in target style (Hu et al., 2017; Dai et al., 2019; Zhang et al., 2018a). Similar to the adversarial learning approach, the learning of the style attribute latent vector is guided using a pre-trained style classifier.

A common key component in the two aforementioned TST approaches is the usage of a style classifier. However, little is known about the effects of the style classifier on these models. For instance, is the style classifier effective in learning the style in the text? What aspects of the text style has the existing style classifier learned? Can the style classifiers distinguish text’s syntax? Can the style classifier guide TST models to generate syntactically correct sentences and in the target style? This paper investigates these questions by conducting an empirical analysis of the style classifiers used in TST models. Extending from our empirical study, we propose the Syntax-Aware Controllable Generation (SACG) model, which includes a syntax-aware style classifier that ensures that the learned style latent representations effectively capture the syntax information for TST. Through extensive experiments with two popular TST datasets and human evaluation, we demonstrated SACG’s ability to outperform the state-of-the-art baselines in the TST tasks.

2 Related Work

In recent years, studies on text style have attracted not only the linguist’s attention but also that of many computer science researchers. Specifically,
computer science researchers are investigating the Text Style Transfer (TST) task that aims to change the text’s stylistic properties while retaining its style-independent content. The recent comprehensive survey (Hu et al., 2020) summarizes the existing TST approaches.

Among these approaches, a popular line of research aims to infer a latent representation for an input sentence and manipulate the generated sentence’s style based on this learned latent representation. Two techniques are commonly used to learn and manipulate the text’s style latent representations: (1) adversarial learning and (2) attribute-controlled generation. Shen et al. (2017) leverages an adversarial training scheme where a classifier is used to evaluate if an encoder is able to generate a latent content representation devoid of style. The text content latent representation is subsequently used to generate a specific style sentence using a style-dependent decoder. Similar works have been proposed where a classifier is pretrained to enable the adversarial learning process in TST models (Zhao et al., 2018; Fu et al., 2018; Chen et al., 2018; Logeswaran et al., 2018; Yin et al., 2019; Lai et al., 2019; Vineet et al., 2019).

Hu et al. (2017) proposed an attribute-controlled generation text style transfer model that utilized a Variational Autoencoder (VAE) (Kingma and Welling, 2013) to learn a sentence’s latent representation $z$ and leverage a style classifier to learn a style attribute vector $s$. Subsequently, $z$ and $s$ are input into a decoder to generate a target style sentence. Similar attribute-controlled generation methods have been proposed for the TST task (Dai et al., 2019; Zhang et al., 2018a; Li et al., 2019).

In the aforementioned methods, pretrained style classifiers played a vital role in guiding the TST task. However, these style classifiers are often pretrained without considering the syntax of sentences. We postulate that syntax is an important aspect of text style, especially in text formality style transfer. This paper empirically demonstrates the importance of modeling syntax in the TST task and proposes a novel syntax-aware TST method that outperforms state-of-the-art TST methods.

### 3 Empirical Study

Before presenting our proposed method, we first conduct an empirical study on the style classifiers used in existing TST methods. The goal is to examine the style classifiers’ ability to learn the syntax style information in a given text.

TextCNN (Kim, 2014), RNN (Cho et al., 2014), and Transformer (Vaswani et al., 2017) are popular style classifiers used in many TST models (Dai et al., 2019; Vineet et al., 2019; Luo et al., 2019; Li et al., 2019; Zhang et al., 2018b). In this study, we train the three style classifiers on GYAFC (Rao and Tetreault, 2018), which is a popular formality transfer dataset used in many TST studies. We first train and test the classifiers using the original GYAFC training and test set. Next, we perturb the sentence structure of the text in the GYAFC test set by disordering the sentences’ word order. The underlying intuition is that there should be syntax differences between formal and informal sentences, and the style classifiers should be able to learn the syntactic style information. Therefore, perturbing the test set’s sentence structure should worsen classification accuracy as the syntactic information in the text is corrupted.

The empirical experiment results show that syntax plays a crucial role in text’s formality. Table 1 shows the results of our empirical experiments. We observed a small 2.9% decrease in style classification accuracy in the disordered test set compared to the original GYAFC test set. We further examined the style classifiers’ performance in different classes. We noted that the classification accuracy for formal sentences sharply decreased as we disordered the test sentences’ word order. However, such observations are not made for informal sentences; the classification accuracy remained fairly constant even when word order was disrupted in informal sentences. From the observations, we postulate that the style classifiers may have focused on the attribute words to predict the style of sentences while neglecting the syntactic information in their style predictions. Furthermore, the style classifiers may have regarded the perturbed sentences as in-
formal ones. Nevertheless, the syntax of informal sentences should be different from the perturbed sentences. The similar classification performance on perturbed sentences demonstrated the style classifiers’ ineffectiveness in capturing different formality styles’ syntax information. More importantly, the style classifier’s inability to learn syntax information could misguide the TST model’s decoder to generate fragmented sentences, especially when transferring sentences to the informal style.

4 Methodology

This section proposes the Syntax-Aware Controllable Generation (SACG) model, which addresses the ineffectiveness of existing TST methods in handling sentence structure when transferring text style. We first introduce Graph Convolutional Networks (GCNs). Subsequently, we explain how the GCNs are utilized to extract sentence structure information in our syntax-classifier and syntax-encoder, which are the two main components in our proposed SACG model. Finally, we describe the learning process of our SACG model.

4.1 GCN and Sentence Structure Representation

As a variant of convolutional neural networks (LeCun et al., 1998), graph convolutional networks (GCNs) (Kipf and Welling, 2017) is designed for graph data and it has demonstrated effectiveness in modeling text data via syntactic dependency graphs (Marcheggiani and Titov, 2017). Consider a graph $\mathcal{G} = \{V,E\}$ where $V$ (where $|V| = n$ is the number of vertices in $\mathcal{G}$) is the set of graph node and $E$ is the set of graph edges. Given a feature matrix $X \in \mathbb{R}^{n \times d}$, where row $x_i \in \mathbb{R}^d$ corresponds to a feature for vertex $i$, the propagation rule of a GCN is given as

$$H^{(l+1)} = \sigma(AH^{(l)}W^{(l)}),$$ (1)

where $H^{(l)} \in \mathbb{R}^{n \times d_l}$ is the feature matrix of the $l$-th layer and $d_l$ is the number of features for each node in the $l$-th layer. $H^{(0)} = X$, $W^{(l)}$ is the weight matrix between the $l$-th and $(l+1)$-th layers, $A \in \mathbb{R}^{n \times n}$ is the adjacency matrix associated with the graph $\mathcal{G}$, and $\sigma(\cdot)$ is a non-linear activation function, such as ReLU or Leaky ReLU. In essence, a GCN takes in a feature matrix $X$ as an input and extract a latent feature matrix $H^{(L)}$ as the output, where $L$ is the number of layers in GCN.

Figure 1 shows the architecture of our proposed syntax-aware style classifier. Our goal is to extract and utilize sentence structure information to guide our SACG model to generate more plausible sentences. The syntactic relations between words in a sentence can be represented using dependency trees (Marcheggiani and Titov, 2017). A dependency tree can be regarded as a directed graph, and the GCNs can be used to extract the latent representation of sentence structure from the dependency trees. Previous studies have attempted to use GNCs to learn syntactic representation from dependency trees (Marcheggiani and Titov, 2017; Bastings et al., 2017). However, many of these existing techniques are over-parameterized, especially on huge datasets. To overcome this limitation, we employ a simpler approach where an adjacency matrix incorporated with directions is used to represent a sentence’s structure. Specifically, the adjacency matrix $A$ is used to represent the dependency relations of all words in the sentence. The column words are head words, and the row words are dependents. We set the element $A_{ij}$ to 1 if there is a dependency between the $i$-th word (head) and the $j$-th word (dependent). Similar to (Marcheggiani and Titov, 2017), we add a self-loop for each node in the graph, where all diagonal elements of $A$ are set to 1.

4.2 Syntax-Aware Style Classifier

In this subsection, we propose syntax-aware style classifier $D$ to encode the syntactic information from the dependency trees better.

Figure 1 shows the architecture of our proposed syntax-aware style classifier. We first encode the to-
Figure 2: Framework of the Syntax-Aware Controllable Generation (SACG) model.

kens in a sentence of size \( n \) as \( s = \{w_1, \ldots, w_n\} \) in the word embedding layer, where \( w_i \) is the \( i \)-th step input of Bi-LSTM. GCN has a limitation in capturing dependencies between nodes far away from each other in the graph. Therefore, instead of performing the graph convolution on the static word embeddings, we perform the GCN operations on top of the Bi-LSTM hidden states (Marcheggiani and Titov, 2017). As such, the GCN will only need to model the relationships for fewer hops. The Bi-LSTM states \( H_{lstm} = \{h_{lstm,1}, \ldots, h_{lstm,n}\} \) serve as input \( x_i = h_{lstm,i} \) to GCN, where \( h_{lstm,i} \) is the concatenation of the forward and backward hidden states. We feed the hidden states into a \( L \)-layer GCN to obtain the hidden representations of each token, which are directly influenced by its neighbors no more than \( L \) edges apart in the dependency tree. Formally, the hidden representation of node \( i \) at the \((l + 1)\)-th layer of GCN is computed by the following equation:

\[
    h^{(l+1)}_i = \sigma \left( \sum_{j=1}^{n} A_{ij} W^{(l)} h^{(l)}_j + b^{(l)} \right) 
\]

where \( A \) is the adjacency matrix of dependency tree, \( W^{(l)} \) and \( b^{(l)} \) are the model parameters, and \( \sigma \) is an activation function. We obtain the hidden representation \( h^{(L)}_i \) of node \( i \) after \( L \) GCN layers.

We noted that some node representations are more informative by gathering information from syntactically related neighbors through GCN. Thus, we utilize scaled dot-product attention (Vaswani et al., 2017) and averaging to aggregate the node representations to sentence representation:

\[
    \text{Attention}(Q, K, V) = \text{softmax}\left( \frac{QK^T}{\sqrt{d_k}} \right)V
\]

where \( Q, K, V \) represent queries, keys, and values, respectively, \( \frac{1}{\sqrt{d_k}} \) is the scaling factor. In practice, we feed the output \( H^{(L)} \) of GCN to \( Q, K, V \). Finally, we obtain the style prediction by feeding the sentence representation into a fully connected neural network followed by the softmax operation.

4.3 Syntax-aware Controllable Generation

Figure 2 shows the framework of our proposed Syntax-Aware Controllable Generation (SACG) model. For each input sentence \( s \) with attribute \( y_o \) and the corresponding adjacency matrix \( A \), the syntax-aware encoder \( E \) encodes \( s \) to a latent representation \( z = E(s, A) \). \( E \) is designed to extract sentence structure using the feature extractor of our proposed syntax-aware classifier. Subsequently, a decoder \( G \) decodes transferred sentence \( \tilde{s} = G(z, y_t) \) or input sentence \( s = G(z, y_o) \) based on the attribute controlling code \( y_t \) or \( y_o \). We employ the Stanford neural dependency parser Stanza (Zhang et al., 2020) to generate the dependency tree for transferred sentences, and the corresponding adjacency matrix \( \tilde{A} \). The transferred sentence \( \tilde{s} \) and the corresponding adjacency matrix serve as the input of the syntax-aware classifier \( D \), and the classifier will evaluate if the transferred sentence has the desired style.

We train the SACG model with classification loss \( L_{cla} \) and reconstruction loss \( L_{tree} \).

Classification Loss \( L_{cla} \): The classification loss ensures the transferred sentence is in the target style. To this end, we apply the pretrained syntax-aware classifier to guide the updates of related parameters such that the output sentence is predicted to be in the target style:

\[
    L_{cla} = -\mathbb{E}_{(s, y_o) \sim D}[\log P(y_t | G(\tilde{s}), \tilde{A})] 
\]

where \( G(\tilde{s}) \) denotes a soft generated sentence based on Gumbel-Softmax distribution (Jang et al., 2017a) and the representation of each word is
defined as the weighted sum of word embeddings with the prediction probability at the current timestep. \( A \) denotes the corresponding adjacency matrix of transferred sentence \( \hat{s} \).

**Reconstruction Loss** \( L_{rec} \): The reconstruction loss attempts to preserve the original content information in the transferred sentences. Specifically, the loss function constrains the model to capture informative features to reconstruct the original sentence using the learned representations. Formally, we define \( L_{rec} \) as follows:

\[
L_{rec} = -\log P(s|z, y_o) \tag{5}
\]

Where \( z \) denotes the hidden representation extracted by our syntax-aware encoder, and \( y_o \) denotes the original style of input sentence \( s \).

**Putting them together**, the final joint training loss \( L \) is as follows:

\[
L = L_{rec} + \lambda L_{cla} \tag{6}
\]

Where \( \lambda \) is a balancing hyper-parameter to ensure that the transferred sentence has the target style while preserving the original content.

5 Experiments

5.1 Experiment Setting

**Datasets.** We evaluate our model on two popular style transfer tasks: (1) Sentiment transfer, and (2) formality transfer. The representative Yelp \(^2\) restaurant reviews dataset (Shen et al., 2017) is selected for the sentiment transfer task. Following the same data preprocessing steps proposed in (Shen et al., 2017), reviews with a rating above 3 are considered positive, and those below 3 are negative. We adopt the same train, development, and test split as (Shen et al., 2017). Rao et al. (2018) released the GYAFCC \(^3\) (Grammarly’s Yahoo Answers Formality Corpus) dataset to facilitate the formality style transfer task. We adopt the Family&Relationship (F&R) domain data for our experiments. Although it is a parallel dataset, the alignments are only used for evaluation and not for model construction. Table 2 shows the training, validation, and test splits of the Yelp and GYAFCC datasets used in our experiments.

**Baselines.** We benchmark SACG against 12 state-of-the-art TST models: \( \text{ARAE} \) (Zhao et al., 2018), \( \text{DualRL} \) (Luo et al., 2019), \( \text{DAST}, \text{DAST-C} \) (Li et al., 2019), \( \text{PFST} \) (He et al., 2020), \( \text{DRLST} \) (Vineet et al., 2019), \( \text{DeleteOnly}, \text{Template}, \text{Del}&\text{Retri} \) (Li et al., 2018), \( \text{DIRR} \) (Liu et al., 2021), and \( \text{HPAY} \) (Kim and Sohn, 2020).

**Training.** The experiments were performed on an Ubuntu 18.04.4 LTS system with 24 cores, 128 GB RAM, and Nvidia RTX 2080Ti. The word embeddings of 300 dimensions are learned from scratch. We use a single Bi-LSTM layer followed by 2 GCN layers. The hidden dimension of the latent representation \( z \) is set to 500, and the learnable vectors with 200 dimensions represent the style labels. The decoder is initialized by a concatenation of the latent representation \( z \) and attribute controlling code \( y \). The syntax-aware style classifier is pretrained for evaluation and guiding the decoder’s generation. After pretraining, the parameters of the classifier are fixed. We use the Gumbel-softmax to back-propagate the loss through discrete tokens from the classifier to the encoder-decoder model (Jang et al., 2017b). We empirically set the learning rate to \( 1 \times 10^{-5} \) and the balancing parameter \( \lambda \) to 1.

5.2 Automatic Evaluation

We evaluate the proposed model and baselines on three criteria commonly used in TST studies: transfer strength, content preservation, and fluency.

**Transfer strength.** A TST model’s transfer strength or its ability to transfer text style is commonly measured using style transfer accuracy (Hu et al., 2020). A syntax-aware style classifier is first pre-trained to predict the style label of the input sentence. The classifier is subsequently used to approximate the style transfer accuracy of the sentences’ transferred style by considering the target style as the ground truth.

**Content preservation.** To quantitatively measure the amount of original content preserved after the style transfer operation, we employed four metrics used in previous work (Fu et al., 2018; Vineet et al., 2019; He et al., 2020):

- **BLEU:** The BLEU score (Papineni et al., 2002) is used to compare the style transferred sentences with the human references provided.

\[\text{Table 2: Dataset statistics for Yelp and GYAFCC.}\]

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{Dataset} & \text{Attributes} & \text{Train} & \text{Dev} & \text{Test} \\
\hline
\text{Yelp} & \text{Positive} & 267K & 38K & 76K \\
& \text{Negative} & 176K & 25K & 50K \\
\hline
\text{GYAFCC} & \text{Informal} & 51K & 2.7K & 1.3K \\
& \text{Formal} & 51K & 2.2K & 1K \\
\hline
\end{array}
\]

\(^2\)https://github.com/shentianxiao/language-style-transfer
\(^3\)https://github.com/raosudha89/GYAFCC-corpus
<table>
<thead>
<tr>
<th>Model</th>
<th>ACC(%)</th>
<th>BLEU</th>
<th>CS</th>
<th>WO</th>
<th>PPL</th>
<th>G-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARAE (Zhao et al., 2018)</td>
<td>76.2</td>
<td>2.2</td>
<td>0.903</td>
<td>0.042</td>
<td>35</td>
<td>0.71</td>
</tr>
<tr>
<td>DeleteOnly (Li et al., 2018)</td>
<td>18.7</td>
<td>16.2</td>
<td>0.945</td>
<td>0.431</td>
<td>74</td>
<td>1.11</td>
</tr>
<tr>
<td>Template (Li et al., 2018)</td>
<td>44.7</td>
<td>19.0</td>
<td>0.943</td>
<td>0.509</td>
<td>102</td>
<td>1.32</td>
</tr>
<tr>
<td>Del&amp;Retri (Li et al., 2018)</td>
<td>50.7</td>
<td>11.8</td>
<td>0.934</td>
<td>0.345</td>
<td>74</td>
<td>1.21</td>
</tr>
<tr>
<td>DualRL (Luo et al., 2019)</td>
<td>59.8</td>
<td>18.8</td>
<td>0.944</td>
<td>0.447</td>
<td>266</td>
<td>1.12</td>
</tr>
<tr>
<td>DAST (Li et al., 2019)</td>
<td>78.3</td>
<td>14.3</td>
<td>0.934</td>
<td>0.350</td>
<td>352</td>
<td>1.01</td>
</tr>
<tr>
<td>DAST-C (Li et al., 2019)</td>
<td>79.2</td>
<td>13.8</td>
<td>0.927</td>
<td>0.328</td>
<td>363</td>
<td>0.98</td>
</tr>
<tr>
<td>Del&amp;Retri (Li et al., 2019)</td>
<td>50.8</td>
<td>11.8</td>
<td>0.909</td>
<td>0.342</td>
<td>31</td>
<td>1.06</td>
</tr>
<tr>
<td>DRLST (Vineet et al., 2019)</td>
<td>59.8</td>
<td>18.8</td>
<td>0.944</td>
<td>0.447</td>
<td>266</td>
<td>1.12</td>
</tr>
<tr>
<td>PFST (He et al., 2020)</td>
<td>48.8</td>
<td>16.5</td>
<td>0.940</td>
<td>0.393</td>
<td>116</td>
<td>1.25</td>
</tr>
<tr>
<td>HPAY (Kim and Sohn, 2020)</td>
<td>43.1</td>
<td>10.4</td>
<td>0.942</td>
<td>0.418</td>
<td>92</td>
<td>1.17</td>
</tr>
<tr>
<td>DIRR (Liu et al., 2021)</td>
<td>71.8</td>
<td>18.2</td>
<td>0.942</td>
<td>0.451</td>
<td>145</td>
<td>1.28</td>
</tr>
<tr>
<td>SACG (ours)</td>
<td>84.1</td>
<td>21.1</td>
<td>0.962</td>
<td>0.591</td>
<td>73</td>
<td>1.69</td>
</tr>
<tr>
<td>Human0</td>
<td>84.6</td>
<td>24.6</td>
<td>0.942</td>
<td>0.393</td>
<td>24</td>
<td>2.00</td>
</tr>
<tr>
<td>Human1</td>
<td>83.8</td>
<td>24.3</td>
<td>0.931</td>
<td>0.342</td>
<td>27</td>
<td>1.89</td>
</tr>
<tr>
<td>Human2</td>
<td>83.6</td>
<td>24.6</td>
<td>0.932</td>
<td>0.354</td>
<td>27</td>
<td>1.91</td>
</tr>
<tr>
<td>Human3</td>
<td>82.1</td>
<td>24.7</td>
<td>0.931</td>
<td>0.354</td>
<td>27</td>
<td>1.90</td>
</tr>
</tbody>
</table>

Table 3: Performance of models on GY AFC dataset (Formality Transfer Task).

<table>
<thead>
<tr>
<th>Model</th>
<th>ACC(%)</th>
<th>self-BLEU</th>
<th>CS</th>
<th>WO</th>
<th>PPL</th>
<th>G-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARAE (Zhao et al., 2018)</td>
<td>83.2</td>
<td>18.0</td>
<td>0.874</td>
<td>0.270</td>
<td>79</td>
<td>1.35</td>
</tr>
<tr>
<td>DeleteOnly (Li et al., 2018)</td>
<td>84.2</td>
<td>28.7</td>
<td>0.893</td>
<td>0.501</td>
<td>130</td>
<td>1.53</td>
</tr>
<tr>
<td>Template (Li et al., 2018)</td>
<td>78.2</td>
<td>48.1</td>
<td>0.850</td>
<td>0.603</td>
<td>250</td>
<td>1.50</td>
</tr>
<tr>
<td>Del&amp;Retri (Li et al., 2018)</td>
<td>88.1</td>
<td>30</td>
<td>0.897</td>
<td>0.464</td>
<td>88</td>
<td>1.66</td>
</tr>
<tr>
<td>DualRL (Luo et al., 2019)</td>
<td>79.0</td>
<td>58.3</td>
<td>0.970</td>
<td>0.801</td>
<td>117</td>
<td>1.98</td>
</tr>
<tr>
<td>DAST (Li et al., 2019)</td>
<td>90.7</td>
<td>49.7</td>
<td>0.961</td>
<td>0.705</td>
<td>181</td>
<td>1.76</td>
</tr>
<tr>
<td>DAST-C (Li et al., 2019)</td>
<td>93.6</td>
<td>41.2</td>
<td>0.933</td>
<td>0.560</td>
<td>274</td>
<td>1.49</td>
</tr>
<tr>
<td>DRLST (Vineet et al., 2019)</td>
<td>91.2</td>
<td>7.6</td>
<td>0.904</td>
<td>0.484</td>
<td>65</td>
<td>1.36</td>
</tr>
<tr>
<td>PFST (He et al., 2020)</td>
<td>85.3</td>
<td>41.7</td>
<td>0.902</td>
<td>0.527</td>
<td>94</td>
<td>1.78</td>
</tr>
<tr>
<td>HPAY (Kim and Sohn, 2020)</td>
<td>86.5</td>
<td>31.2</td>
<td>0.886</td>
<td>0.450</td>
<td>85</td>
<td>1.66</td>
</tr>
<tr>
<td>DIRR (Liu et al., 2021)</td>
<td>94.2</td>
<td>52.6</td>
<td>0.957</td>
<td>0.715</td>
<td>292</td>
<td>1.63</td>
</tr>
<tr>
<td>SACG (ours)</td>
<td>93.0</td>
<td>57.7</td>
<td>0.971</td>
<td>0.778</td>
<td>74</td>
<td>2.23</td>
</tr>
</tbody>
</table>

Table 4: Performance of models on Yelp dataset (Sentiment Transfer Task).

in the GYAF dataset.

• **self-BLEU**: The self-BLEU score is adopted by comparing the style transferred sentence with its original sentence. This metric is used when human reference is not available.

• **Cosine Similarity**: Fu et al. (2018) calculated the cosine similarity between original sentence embedding and transferred sentence embedding. The two sentences’ embeddings should be close to preserve the semantics of the transferred sentences.

• **Word Overlap**: Vineet et al. (2019) employed a simple metric that counts the unigram word overlap rate of the original and style transferred sentences.

**Fluency.** Generating fluent sentences is a common goal for most natural language generation models. GPT-2 (Radford et al., 2019) is a large-scale transformer-based language model that is pre-trained on large text corpus. We fine-tuned GPT-2 on the GYAF and Yelp datasets and use the model to measure the perplexity (PPL) of transferred sentences. The sentences with smaller PPL scores are considered more fluent.

**Geometric Mean (G-Score)**: We compute the geometric mean of ACC, self-BLEU, BLEU, CS, WO and 1/PPL. Notably, we take the inverse of the calculated perplexity score because a smaller PPL score corresponds to better fluency.

### 5.2.1 Automatic Experiment Results

Table 3 shows the performance of the proposed SACG model and baselines on the formality transfer task. SACG has achieved the best G-Score, outperforming the state-of-the-art baselines. Nevertheless, we noted that none of the TST models could score well on all evaluation metrics. Many of the baselines can only perform well on transfer strength or content preservation, but not on both evaluation criteria. SACG has outperformed the baselines in G-Score, and achieve 84.1% transfer accuracy and 21.1 average BLEU score. The GYAF dataset also provided the performances of four human references performing the formality transfer task on the test set. The BLEU score of
Table 5: Human evaluation results on GYAFC dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Style(%)</th>
<th>Content</th>
<th>Fluency</th>
</tr>
</thead>
<tbody>
<tr>
<td>DualRL</td>
<td>28.5</td>
<td>4.09</td>
<td>4.52</td>
</tr>
<tr>
<td>DAST</td>
<td>27.5</td>
<td>3.22</td>
<td>3.68</td>
</tr>
<tr>
<td>PFST</td>
<td>24.0</td>
<td>3.91</td>
<td>4.54</td>
</tr>
<tr>
<td>Del&amp;Retri</td>
<td>25.5</td>
<td>2.61</td>
<td>3.23</td>
</tr>
<tr>
<td>SACG</td>
<td>44.5</td>
<td>4.39</td>
<td>5.07</td>
</tr>
</tbody>
</table>

Table 6: Average Tree Edit Distance (TED) of constituency tree between TST model generated sentences and 4 human references in GYAFC.

<table>
<thead>
<tr>
<th>Model</th>
<th>TED</th>
<th>Model</th>
<th>TED</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRLST</td>
<td>19.2</td>
<td>DeleteOnly</td>
<td>18.2</td>
</tr>
<tr>
<td>ARAE</td>
<td>18.1</td>
<td>Template</td>
<td>17.9</td>
</tr>
<tr>
<td>DualRL</td>
<td>15.2</td>
<td>Del&amp;Retri</td>
<td>21.0</td>
</tr>
<tr>
<td>DAST</td>
<td>16.6</td>
<td>HPAY</td>
<td>18.4</td>
</tr>
<tr>
<td>PFST</td>
<td>15.5</td>
<td>DIRR</td>
<td>15.5</td>
</tr>
<tr>
<td>DAST-C</td>
<td>16.9</td>
<td>SACG (ours)</td>
<td>13.2</td>
</tr>
</tbody>
</table>

Each human reference is calculated with the other three human references. Interestingly, we observe that SACG’s performance on the three TST evaluation criteria is comparable and close to human references’ performance.

Similar results were observed for the sentiment transfer task. Table 4 shows the performance of the proposed SACG model and baselines on the Yelp dataset. We computed the self-BLEU scores as no human references are provided for the Yelp test set. Similarly, SACG outperformed the baselines in G-score. We observe that the average style transfer accuracy in Yelp is 86.3%, which is significantly higher than GYAFC’s average score of 66.0%. The difference in the average style transfer accuracy highlights the challenge of the formality transfer task. We also noted that most models performed better in this task compared to the formality transfer task. Nevertheless, the trade-off phenomenon between transfer strength and content preservation is still observed in the sentiment transfer task.

5.3 Human Evaluation

To further evaluate SACG’s performance in generating syntactically correct sentences in target style, we conducted a human-based evaluation study. Specifically, we first randomly sampled 200 sentences from the GYAFC dataset. Next, we perform text style transfer for the sampled sentences using SACG and four competitive baselines. Finally, we recruited two linguistics researchers (i.e., participants) to evaluate the style-transferred sentences generated by the TST models. The participants are asked to evaluate the generated sentences on the three criteria discussed in the earlier section. Specifically, for Transfer Strength, participants are asked to indicate if the generated sentences are in the target style (i.e., a binary true/false indicator). For Content Presentation, the participants are asked to rate the amount of content preserved in the generated sentences using a 6-point Likert scale. 1: no content presented, and 6: all content are preserved. Similarly, for Fluency, the participants are asked to rate fluency in the generated sentences using a 6-point Likert scale. 1: too many grammatical errors, and 6: perfect and fluent sentence.

To minimize biases, we do not display the models’ names and we shuffled the order of the models when displaying their generated sentence. Therefore, the participants do not know which model generates a particular sentence.

5.3.1 Human Experiment Results

Table 5 shows the human evaluation results. For the transfer style, we compute the models’ style transfer accuracy using the binary feedback from the participants. We compute the models’ average 6-point Likert scores for content preservation and fluency criteria. SACG is observed to outperform the baselines in all three criteria. SACG is also rated to generate more syntactically sound and fluent sentence compared to the baselines. To check for participant bias, we compute the inter-annotator agreement between the participants. The Cohen’s kappa coefficients on style transfer strength, content preservation, and fluency are 0.54, 0.76, and 0.72, respectively. The participants have substantially high agreement on the content presentation and fluency. However, the participants’ agreement for style transfer strength is moderate as text formality is subjective, and the participants are only asked to perform binary indication.

5.4 Syntax Evaluation

As human references are available in the GYAFC dataset, we compare the syntax of the sentences generated by the TST models with the human references. Specifically, we compute the constituency tree edit distance (TED) to measure the syntactic similarity between generated sentences and human references. The intuition is that the TST model that could generate sentences with similar syntactic structure as the human references would likely have learned the syntactic information associated with the text formality style. To compute the constituency TED, we parse the sentences using Stan-
ford CoreNLP and compute the TED between constituency parsing trees.

Table 6 shows the syntax evaluation results. We noted that SACG outperformed the baseline in generating sentences that are syntactically similar to human references. This superior performance in both the formality transfer task and syntax evaluation suggests that SACG is able to learn the syntax information of formal and informal text to perform better text formality transfer.

5.5 Ablation Study

We also conducted an ablation study to further examine the importance of syntax-aware classifier and encoder in the SACG model. Table 7 shows the results of our ablation study. In the “w/o syntax-aware encoder” setting, we replace the syntax-aware encoder with a one-layer GRU (Cho et al., 2014). We noted a small decrease in performance for both formality transfer and sentiment transfer tasks when the encoder is replaced. In the “w/o syntax-aware encoder & classifier” setting, we further replace the syntax-aware classifier with a TextCNN (Kim, 2014) classifier. Interestingly, we observe a sharp decrease in performance for both formality transfer and sentiment transfer tasks. In particular, the absence of the syntax-aware encoder and classifier greatly worsens the fluency of the sentences. Our ablation study noted that the syntax-aware encoder and classifier play vital roles in ensuring SACG generates fluent target-style sentences that preserve the original content.

5.6 Case Study

We conduct some case studies by presenting randomly sampled examples and the corresponding style transferred output of SACG and the top three baselines ranked by G-Score. Table 8 shows the example outputs on the GYAFC and Yelp datasets. For the Yelp dataset, we observe that DualRL, PFST, and SACG are able to transfer the sentiment of the source sentence correctly. The generated sentences are also fluent and have preserved the original content (i.e., going back to a venue). The formality transfer task is observed to be more challenging, as we noted that most of the baselines could not generate acceptable output sentences. The baselines have generated output sentences with grammatical errors, making it harder to judge if the style has been successfully transferred. Albeit the difficulty of the task, SACG is able to generate a fluent sentence that preserved the original content.

6 Ethical Considerations

TST algorithms have many real-world applications. For example, these algorithms can improve target marketing messages’ persuasiveness and integrate into writing tools to improve users’ writing style. However, TST algorithms inherently run the risk of being misused for document forgery, impersonation, and sock-puppeting. To mitigate these risks, we will add access control to our code repository, and we would share our codes after the requester has acknowledged our ethical disclaimer.
7 Conclusion

In this paper, we empirically examined the style classifier used in existing TST models and demonstrated that the existing style classifier could not learn the text syntax effectively. We proposed SACG, a novel deep generative framework that considers syntax when learning style latent representation. We conducted extensive experiments on two benchmark datasets and benchmarked SACG against competitive TST models. The automatic and human-based evaluation experiment results showed that SACG outperforms state-of-the-art methods. Our case studies also demonstrated that SACG is able to generate fluent target-style sentences that preserved the original content. For future work, we will continue to explore other methods to improve the structural representations of text and incorporate them to perform better TST.

Acknowledgement

This research is supported by Living Sky Technologies Ltd, Canada under its research exploratory funding initiatives. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of Living Sky Technologies Ltd, Canada.

References


Transfer learning for Czech Historical Named Entity Recognition

Helena Hubková†‡
† Department of Computer Science and Engineering, University of West Bohemia, Plzeň, Czech Republic
hhubkova@kiv.zcu.cz

Pavel Král†‡
‡ NTIS – New Technologies for the Information Society, University of West Bohemia, Plzeň, Czech Republic
pkral@kiv.zcu.cz

Abstract

Nowadays, named entity recognition (NER) achieved excellent results on the standard corpora. However, big issues are emerging with a need for an application in a specific domain, because it requires a suitable annotated corpus with adapted NE tag-set. This is particularly evident in the historical document processing field.

The main goal of this paper consists of proposing and evaluation of several transfer learning methods to increase the score of the Czech historical NER. We study several information sources, and we use two neural nets for NE modeling and recognition.

We employ two corpora for evaluation of our transfer learning methods, namely Czech named entity corpus and Czech historical named entity corpus. We show that BERT representation with fine-tuning and only the simple classifier trained on the union of corpora achieves excellent results.

1 INTRODUCTION

Recently, named entity recognition (NER) achieved outstanding results on standard NER corpora. Particularly on English CONLL-2003 corpus 90% F-measure has been overcome, which is sufficient for several real applications.

However, big issues are emerging with a need for an application in the specific domain which requires an appropriately annotated corpus including adapted NE tag-set. This issue is particularly evident in the case of historical documents in Czech language on which we focus on the project “Modern Access to Historical Sources”1. Manual annotation of this corpus is very expensive and time consuming. Moreover, the presence of a linguist is necessary. We will use the information about named entities as additional metadata for information retrieval and knowledge extraction.

Transfer learning targets at reusing information obtained from one corpus to improve the results of a model learned on an analogous task with few resources. To overcome this issue, we propose and evaluate several transfer learning approaches to improve the results of the Czech historical NER when the annotated resources are limited. The following information sources are considered and studied for this task:

- pre-trained fastText2 word vectors;
- pre-trained word2vec3 word vectors;
- pre-trained Slavic BERT4 contextual text representation;
- Czech contemporary NE corpus from different domain.

We employ two neural based models, namely recurrent Bidirectional long short-term memory (BiLSTM) (Graves et al., 2005) and Bidirectional encoder representations from transformers (BERT) model (Devlin et al., 2019) with a simple perceptron for NER modelling and recognition. We use two corpora for evaluation of our experiments.

Note that, to the best of our knowledge, this is the first attempt to employ transfer learning in the field of Czech historical NER.

2 RELATED WORK

Rodriguez et al. (Rodriguez et al., 2018) presented reproduction paper focused on transfer learning for entity recognition. They compared seven new corpus pairs results and other researches that were

1http://www.portafontium.eu/
2https://fasttext.cc/
3https://code.google.com/p/word2vec/
4https://github.com/deepmipt/Slavic-BERT-NER
published previously. They showed that if there is a small labelled target dataset, the simpler approaches work better in compare to neural transfer approaches that work better for larger labelled target data set. They reached an F1 score of 71.81% for the period 1814–1817 and 70.35% for the period 1685–1691.

In term of NER in historical texts, different methods have been applied for English so far; rule-based NER (Grover et al., 2008), Maximum entropy Markov model and Conditional random fields (Packer et al., 2010). Different tools for NER (Rodriguez et al., 2012) as OpenNLP, Stanford NER, AlchemyAPI and OpenCalais are available. NER for historical newspapers was researched by Mac Kim and Cassidy (2015) (English), Neudecker Neudecker (2016) (Dutch, French, German) and Kettunen et al. (2016) (Finnish). In case of historical newspapers, Stanford NER (Finkel et al., 2005) was applied to the 155 million OCRed articles from historical Australian newspapers by Sung hwan and Cassidy (Mac Kim and Cassidy, 2015) and they described how the data can be exploited using a clustering method. Moreover, Neudecker (Neudecker, 2016) created an open corpus for NER in Dutch, French and German based on OCRed historical newspapers as part of the Europeana Newspapers project.6 using Stanford NER for German. Similarly, (Kettunen et al., 2016) evaluated NER tools for Finnish using OCRed Finnish historical newspaper collection Digi. Transfer learning for NER was implemented by Lee et al. (Lee et al., 2018), similarly, for historical German NER by Riedl and Padó (2018) and Schweter and Baiter (2019).

Transfer learning for NER was implemented by Lee et al. (Lee et al., 2018) using artificial neural nets for two different datasets of patient note de-identification. They demonstrated that an ANN model trained on large labeled data set could be transferred to get state-of-the-art results on the datasets with small number of labels. Transfer learning for historical NER was investigated by Riedl and Padó (Riedl and Padó, 2018). They compared different NER models and methods for both contemporary German (large datasets) and Historic German (small datasets). They concluded that the best performance has BiLSTM model with a CRF as a top layer if enough data is available. On the other hand, the BiLSTM model using transfer learning showed that it is more effective for small data. They trained the model with large datasets of contemporary German and then they tuned on small historical ones. More recently, Schweter and Baiter (Schweter and Baiter, 2019) applied the contextual string embeddings (Akbik et al., 2018) (Flair) for German Historic NER. They also used synthetic masked language modelling (SMLM) that randomly adds noise during the training in comparison to the masked language modelling in BERT by Devlin et al. (Devlin et al., 2019). They showed that pre-trained models on specific datasets can reach state-of-the-art results in the case of Historic German. However, the SMLM approach showed the second best results. They also experimented with pre-trained fastText embeddings.

Recently, NER for contemporary Czech was researched by Štraka et al. (2019) using BERT (Devlin et al., 2019) and Flair (Akbik et al., 2018). Similarly, Arkhipov et al. (2019) presented multilingual NER for Russian, Bulgarian, Czech and Polish.

In term of NER for contemporary Czech, Štraka et al. (Straka et al., 2019) recently presented their sequence-to-sequence model to evaluate BERT (Devlin et al., 2019) and Flair (Akbik et al., 2018) and their combination on Czech named entity corpus (CNEC) versions 1.1 and 2.0. For CNEC 1.1, they reached 87.62% F1-score using Flair, 89.85% using BERT and 89.91% using both of them. For types of CNEC 2.0., they achieved 81.65% F1-score for Flair, 86.23% for BERT and 85.52% for both.

Moreover, Arkhipov et al. (Arkhipov et al., 2019) presented multilingual named entity recognition in Russian, Bulgarian, Czech and Polish (Asia Bibi datasets from BSNLP 2019 Shared Task) using BERT model and additional word-level CRF layer. This approach reached state-of-the-art results: 93.9 F1 score for Czech, 87.3 for Russian, 87.2 for Bulgarian and 93.2 for Polish, respectively.

In the case of text embeddings, Akbik et al. proposed a pre-trained model of contextual string embeddings (Flair) for NER that considers words as sequences of characters. They experimented with a BiLSTM-CRF model proposed by Huang et al. (Huang et al., 2015) and different approaches to word embeddings. They extended the model by adding a concatenation of pre-trained static word embeddings with contextual ones and a concate-

5http://www.europeana-newspapers.eu/

6https://ufal.mff.cuni.cz/cnec
nation of task-trained character features with contextual string embeddings. They reached 93.09% F1-score for English and 88.32% for German with this model configuration for CoNLL2003 shared task.

3 CORPORAS

We experimented with two corpora: Czech named entity corpus (CNEC) and Czech historical named entity corpus (CHNEC). CNEC corpus contains almost 9,000 sentences and more than 35,000 occurrences of the Czech named entities. The corpus uses two-level NEs annotation scheme and the first-level contains 10 main NE types and the second-level is composed of 62 NE subtypes. To be able to map NEs from CNEC to CHNEC, we use only five NE types from the first annotation level which are same for both corpora.

CHNEC\(^7\) contains 73,647 tokens and 4,017 named entity occurrences. The corpus was created from Czech historical newspaper Posel od Čerchova from second half of 19th century and distinguishes five NE types: Personal names, Institutions, Geographical names, Time expressions and Artifact names/Objects. The corpus is encoded in IOB format (Ramshaw and Marcus, 1995), where \(B\) represents one-token entity or the beginning of multi-token named-entity, \(I\) inside tokens of multi-token named-entity and \(O\) stands for all tokens that are not a named-entities.

4 METHODS

4.1 Models and Representations

4.1.1 BiLSTM with Word-level Embeddings

The first approach uses BiLSTM model with word-level representation of the sentences. We used similar network structure and similar hyperparameters as presented by Hubková et al. (2020) (Table 1) with two different word representation methods: fastText and word2vec.

Note that we use BiLSTM model with randomly initialized word embeddings as a baseline. This approach does not consider any transfer because the embeddings are learned during the training of the network only of the available training data.

4.1.2 BERT with Perceptron

The second method uses BERT model (Devlin et al., 2019) for representation of the text and a simple single-layer perceptron (SLP) with only one softmax layer is used for NE recognition. The main advantage of this approach in comparison with the previous one is that BERT considers the different word meaning when used in the different context.

This model uses unlabeled data to pretrain deep bidirectional representations by jointly conditioning on both left and right context in all layers. Sequences of word tokens (or subtokens) in the sentence are used as an input and the outputs are class probabilities among the classes.

4.2 Transfer Learning

4.2.1 Transfer from Embedding Word Vectors

The first transfer learning approach is focused on the embedding vectors. Our embedding vectors are built using different models, and thus they kept different information. FastText considers word and also sub-word units, therefore it should encode semantic and syntactic information as well. However, word2vec is trained properly using word tokens, hence it includes mostly semantic information.

The initial embeddings are learned on huge unlabeled text corpora coming from different domains and containing different language (contemporary Czech instead of historical one). For that reason, we assume that the further fine-tuning of these embeddings on the target data should improve the final NE recognition score. Therefore, in this approach, we compare fastText and word2vec and embeddings with two scenarios. The first one uses only static embeddings without a subsequent training and the second one tries to adjust these vectors into our task during network training.

4.2.2 Transfer from BERT Representation

The second approach is based on the transfer from BERT representation. In order to have as much precious representation as possible, we use pre-trained Slavic BERT proposed by Arkipiov et al. (2019). This model is based on a multilingual BERT model and fine-tuned with Czech, Russian, Bulgarian and

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>Range</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM state #</td>
<td>[100; 500]</td>
<td>250</td>
</tr>
<tr>
<td>LSTM layer #</td>
<td>[1; 3]</td>
<td>1</td>
</tr>
<tr>
<td>Learning rate</td>
<td>[0.001; 0.01]</td>
<td>0.004</td>
</tr>
<tr>
<td>Epochs</td>
<td>[60;120]</td>
<td>80</td>
</tr>
<tr>
<td>Dropout</td>
<td>[0.25; 0.85]</td>
<td>0.65</td>
</tr>
</tbody>
</table>

Table 1: Overview of hyper-parameter optimization

\(^7\)http://chnec.kiv.zcu.cz/
Polish data. We assume that the further fine-tuning of this representation on the target data will still improve the final NE score. Therefore, in this approach, we perform another fine-tuning by training on our historical data.

4.2.3 Transfer from Different Corpus
The third approach assumes that different NER corpora (see Section 3) should include complementary information and the usage of these together will improve the final score of the target task. The following scenarios are considered, evaluated and compared:

- training only on CHNEC corpus (to show the impact of the approaches below);
- training only on CNEC corpus (to show the results when the target annotated data are not available);
- union of both corpora and training on this large dataset;
- initial training on the CNEC corpus and fine-tuning on the target one (CHNEC).

4.3 Cross-corpus Method
As a cross-corpus method we mean that we trained BiLSTM or BERT model with CNEC training data set (source corpus) and models and we tested on CHNEC test data (target corpus). This approach showed if bigger corpora for contemporary language itself can be used for tagging a smaller historical texts.

We experimented with token classification PyTorch module for NER by Wolf et al. (2019) and we used pre-trained Slavic BERT model.

BERT token classification method has a linear layer on top of the hidden-states output and this model is available through PyTorch module. Pre-trained Slavic BERT is based on BERT model by Devlin et al. (2019) and extended by word-level CRF layer. The model is tuned on four Slavic languages Russian, Bulgarian, Czech and Polish.

5 Evaluation and Results
Table 2 and Table 3 show the results of our experiments. We use the standard precision, recall and macro-averaged F1-score (Powers, 2011) metrics for evaluation. In all cases, we calculate the final score on the testing part of CHNEC corpus. Qualitative analysis in Section 5.1 is based on the observed linguistics phenomena in both development and test data sets.

The first part of Table 2 presents the results of our approaches dealing with the transfer of embedded word vectors. BiLSTM model trained only on CHNEC corpus is used for NE recognition. The results show that fastText representation brings significantly better results than word2vec one. These results further illustrate that the fine-tuning of the embeddings has only a positive impact in the word2vec case, and unfortunately, it does not bring any improvement in the case of the fastText representation. This behaviour should be justified that fastText word representation corresponds better to our task, and our training data are too small and differ from the testing set for the further improvement of the model. Based on these results, we will use for the following experiments only fastText fixed embeddings.

The second part of the table shows the experiments using BiLSTM model with different approaches for training. These results show that it is possible to obtain F-measure 45% using only the different corpus (any additional annotation is not required). Moreover, it has been demonstrated that the transfer from CNEC into CHNEC does not bring any positive impact on the final NER.

The last part of Table 2 shows the results of our transfer learning approaches using BERT representation with the simple single-layer perceptron as a classifier. The impact of the different training approaches is also considered. These results show clearly that BERT representation with fine-tuning and only a simple classifier brings significantly better results that all the approaches evaluated previously. This should be explained by the fact that word context representation is much more accurate than the word-level one. This experiment further illustrates that the additional data coming from another NER corpus is beneficial to improve the final NER score by 2%.

Another observation is that the values of the precision and recall are balanced, however in the previous series of experiments, these values differ significantly.

The previous experiment does not consider the individual NE types. However, this information could be very interesting for further improvement of the training strategy and the model itself. We assume, that the size of the training data and the
Table 2: NER results of the different approaches evaluated on the testing set of CHNEC corpus (best results in bold)

<table>
<thead>
<tr>
<th>No.</th>
<th>Approach</th>
<th>Training data</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BiLSTM (baseline)</td>
<td>CHNEC</td>
<td>0.63</td>
<td>0.52</td>
<td>0.57</td>
</tr>
<tr>
<td></td>
<td>BiLSTM + fastText fixed</td>
<td>CNEC</td>
<td>0.76</td>
<td>0.70</td>
<td>0.73</td>
</tr>
<tr>
<td></td>
<td>BiLSTM + fastText fine-tuned</td>
<td>CNEC → CHNEC</td>
<td>0.65</td>
<td>0.60</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td>BiLSTM + word2vec fixed</td>
<td>CHNEC</td>
<td>0.67</td>
<td>0.45</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td>BiLSTM + word2vec fine-tuned</td>
<td>CNEC</td>
<td>0.61</td>
<td>0.55</td>
<td>0.58</td>
</tr>
</tbody>
</table>

Table 3: Results of the individual NE types using the best model and training scenario: SLP + Slavic BERT fine-tuned on CNEC + CHNEC. The last two columns show the NE numbers in the whole and in the testing part of the CHNEC corpus.

<table>
<thead>
<tr>
<th>NE Class</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
<th>NE # in Total</th>
<th>NE # in Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geographical names</td>
<td>0.91</td>
<td>0.93</td>
<td>0.92</td>
<td>1104</td>
<td>137</td>
</tr>
<tr>
<td>Artifact names/Objects</td>
<td>0.85</td>
<td>0.86</td>
<td>0.86</td>
<td>829</td>
<td>101</td>
</tr>
<tr>
<td>Time expressions</td>
<td>0.79</td>
<td>0.75</td>
<td>0.77</td>
<td>506</td>
<td>55</td>
</tr>
<tr>
<td>Personal names</td>
<td>0.72</td>
<td>0.76</td>
<td>0.74</td>
<td>1292</td>
<td>91</td>
</tr>
<tr>
<td>Institutions</td>
<td>0.57</td>
<td>0.63</td>
<td>0.59</td>
<td>286</td>
<td>37</td>
</tr>
</tbody>
</table>

The learning approaches on the randomly selected sample containing about 100 randomly selected sentences per model and approach.

If we train BiLSTM model only with CNEC data, the model showed that this method correctly tagged especially geographic names and personal names, e.g.: names of towns such as Prahy or Horšovský Týn; names of persons such as Vojtěcha Bittmara or Petr Bedřich Florian. The time expressions were tagged correctly in case that the format of the time in CNEC was similar to format in CHNEC, e.g. 20 . února 1775 (“20th February 1775”). Similarly, the named entity Sokol (name of a sport institution) was tagged correctly as this named entity occurs in both CNEC and CHNEC. However, the other NE types were rather erroneous.

On the other hand, if we compare these results with a basic BiLSTM model that was trained only on in-domain historical data, we can see that the specific language expressions that occur in historical CHNEC texts are not tagged. CHNEC corpus contains a number of abbreviations, e.g. 26 . června t . r . (“26th June this year”) or c . k . okresní finanční ředitelství v Plzni (“Imperial-Royal District Financial Directorate in Pilsen”). More-
over, using dots in the named entities in this corpus is inconsistent, it means that it contains both 29. June and 29. June or name of person Jos. Kralovec.

Then, we analyze the results of the approaches which use BERT representation. Generally, BERT improves the results with its ability to correctly tag NEs that do not occur in the training data. As we fine-tuned the Slavic BERT model using the combination of CNEC and CHNEC, the model overcomes the problems with abbreviations mentioned above. However, some false positive cases occur as well, e.g. c. k. okresní hejtman ("Imperial-Royal district governor") was tagged as Institution.

Geographical names that occur more frequently in CHNEC than most other NE types reached 0.92 F1-score in comparison to Institutions that occur only 286 times in the whole CHNEC and reached 0.59 F1-score.

Next to the pure occurrences in the corpora, the names of the Institutions differ between historical and contemporary language a lot as many institutions do not exist in the present anymore, e.g. spořitelný kr. města Domažlic ("savings bank of the royal town of Domažlice"). From this point of view, Time expressions or Personal names are more stable in time. The Institution are also usually long multi-word expressions (e.g. "všeobecné úvěrní a obchodní banky" ("general credit and commercial bank") in contrast to shorter and more consistent Time expressions and Personal names. This fact also complement the previous justifications of the lowest recognition score for the class Institutions.

6 Conclusions

In this paper, we proposed and evaluated several transfer learning approaches in order to improve the results of the Czech historical NER. We considered and studied the following information sources: pre-trained fastText and word2vec word representations, pre-trained Slavic BERT contextual text representation and another NE corpus from different domain. We used two popular models, namely recurrent BiLSTM and BERT with a simple perceptron for NER modelling and recognition. We have shown that fastText representation gives significantly better results than word2vec model. It has been also demonstrated that the transfer from CNEC into CHNEC with BiLSTM model does not improve the final NER score. We have further presented that BERT representation with fine-tuning and only the simple classifier trained on the union of corpora brings the best results (F-measure 82%). Based on the analysis of the errors, we can also conclude that the combination of the different models/approaches would not bring any further improvement.

Acknowledgements

This work has been partly supported from ERDF “Research and Development of Intelligent Components of Advanced Technologies for the Pilsen Metropolitan Area (InteCom)” (no.: CZ.02.1.01/0.0/0/17_048/0007267) and by Grant No. SGS-2019-018 Processing of heterogeneous data and its specialized applications.

References


Personality Trait Identification
Using the Russian Feature Extraction Toolkit

James R. Hull*, Valerie Novak*, C. Anton Rytting*,
Paul Rodrigues†,*, Victor M. Frank*, and Matthew Swahn†

*University of Maryland, College Park, MD, USA
†Accenture, Washington, DC, USA
{jhull1, vnovak, crytting, prr, vmfrank}@umd.edu, matthew.swahn@accenture.com

Abstract
Feature engineering is an important step in classical NLP pipelines, but machine learning engineers may not be aware of the signals to look for when processing foreign language text. The Russian Feature Extraction Toolkit (RFET) is a collection of feature extraction libraries bundled for ease of use by engineers who do not speak Russian. RFET’s current feature set includes features applicable to social media genres of text and to computational social science tasks. We demonstrate the effectiveness of the tool by using it in a personality trait identification task. We compare the performance of Support Vector Machines (SVMs) trained with and without the features provided by RFET; we also compare it to a SVM with neural embedding features generated by Sentence-BERT.

1 Introduction
Data scientists and computer scientists working on natural language processing (NLP) problems are occasionally confronted with tasks in languages they do not know well. Technical linguistic issues that may be well known to researchers and language practitioners familiar with the language can be quite important to effective language engineering, and can take a lengthy career to master.

The Russian Feature Extraction Toolkit (RFET) unites a suite of tools for Russian language processing that would allow a programmer unfamiliar with the language the ability to get started quickly in common text classification tasks such as sociolinguistic factor classification, sentiment classification, and emotion analysis. Additionally, this toolkit provides those advanced in Russian NLP convenient accessibility of features and feature combinations to quickly iterate through multiple experiment scenarios.

Deep learning approaches are showing great promise, and feature extraction is less commonly utilized in these approaches. Still, there are existing systems in production which harness classical machine learning techniques. RFET, and tools like it for other languages, could improve performance of these systems without a complete redesign. Further, there are many languages in which deep learning approaches still lack pre-trained models or even a sufficient quantity of text examples to create them. The methodology presented in this paper could be used to design Feature Extraction Toolkits for these languages, in these cases.

This toolkit does not replace feature extraction functions in libraries like NLTK (Bird, 2006) or scikit-learn (Pedregosa et al., 2011). These systems can produce language-independent statistical measurements on text that could be used as features. It also does not seek to replace toolkits for basic NLP pipeline elements, such Natasha,¹ though there is some overlap between the tools. This toolkit focuses on uniting Russian language resources that provide linguistically-informed features that are distinctive of the Russian language or otherwise not adequately represented in features used in text analytics or corpus linguistics, with a focus on social media Russian. The toolkit includes 70 features.

In order to validate the toolkit’s utility, we evaluate its efficacy in a challenging task with little prior work in Russian: namely, the identification of personality traits from user-generated social media text. This task is further described in Section 2.2.

¹https://github.com/natasha/natasha

588
Proceedings of Recent Advances in Natural Language Processing, pages 588–597
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_067
Section 3 describes RFET’s collected features in more detail; Section 4 describes the dataset and other methodological details of the task and our results. Other sections outline larger context and limitations of the study.

2 Related Work

2.1 Language-specific Feature Extraction

Structured Programming for Linguistic Cue Extraction (SPLICE) (Moffitt et al., 2012) is a tool for English language feature extraction. SPLICE offers (via API\(^2\)) a variety of lexicons and other features relevant to credibility assessment and deception detection, including lexicons for deference, positive and negative self-evaluation, affect and sentiment (from SentiWordNet). SPLICE’s features include part of speech, verb tense and passive voice (for immediacy), spoken word counts for hedging and disfluency, and a variety of readability scores. It offers a mechanism for users to submit their own lexicons.

The Arabic Data Science Toolkit (ADST) (Rodrigues et al., 2018) is a Python toolkit for analyzing Arabic, particularly social media Arabic. It addresses features both in Modern Standard Arabic and Egyptian Colloquial Arabic, and focuses on features that highlight emotion, such as laughter and emoji, and informal expressions of intensity, such as elongated words. It also includes several language-specific lexicons, such as honorifics, polite and pious expressions, abusive language, and transitional phrases. Its coverage of emotional language is somewhat incomplete, focusing more on positive emotions such as happiness and humor than negative emotions.

Linguistic Inquiry and Word Count (LIWC) (Pennebaker, 1993; Pennebaker et al., 2007, 2015) is a collection of vetted lexicons focused mainly on lexical features of psychological interest, though it also covers more general linguistic categories as well. All categories are vetted multiple times by human judges. The 2015 version has been updated and expanded to include “netspeak” language found in social media and SMS, including some common informal abbreviations and emoticons. Developed originally for English, it has been ported to 12 languages, including Russian. It is available only under paid license (even for non-commercial academic use) and (for Russian) only as a stand-alone program. The API only supports English.

Another tool, Empath (Fast et al., 2016), might be characterized as a partially automated extension of LIWC, with a framework for further extensibility. Empath uses a combination of human-generated seed words, semantic embedding-based term discovery to grow topic lexicons (categories) from these seed terms, and crowd-powered filtering to validate these categories. Unlike most embedding-based models, it is trained largely on fiction works in the public domain, which are claimed to offer more general coverage than other domains. It offers many more topics and categories than LIWC, but novel categories are largely unvetted. The pre-validated models currently available are (to the best of our knowledge) only available in English.

Natasha and DeepPavlov\(^3\) (Burtsev et al., 2018) are NLP pipeline toolkits specifically built for Russian. They bring together NLP tools such as word token and sentence segmentation, word embeddings, morphological syntactic tools, and NER. Natasha adds fact extraction; DeepPavlov various conversational agent functions. As general purpose tools, they complement RFET, which has a more specific focus on social media text.

2.2 Personality Trait Identification

While several personality taxonomies exist, the most well researched set of personality traits is called the Five Factor Model, Big Five, or OCEAN (Openness, Conscientiousness, Extraversion, Agreeableness, and Negative Emotionality or “Neuroticism”), which is typically measured via a 60-item self-report questionnaire, the Big Five Inventory-2 (BFI-2) (Soto and John, 2017). The BFI-2 has been translated into Russian and validated with samples of students and internet users (Shchebetenko et al., 2020).

Golbeck et al. (2011a,b) were among the first to examine the efficacy of inferring personality from user-generated social media text (from Twitter and Facebook). Since that time, a number of studies have followed suit. Farnadi

\(^2\)http://splice.cmi.arizona.edu/

\(^3\)https://deeppavlov.ai/
et al. (2016) conducted a comparative analysis of computational methods on English social media text from three separate platforms: Facebook, YouTube, and Twitter. While most work in social media personality trait identification has focused on English, other European languages have also been examined. For example, the PAN research group organized a shared task in 2015 with Twitter data from four languages: English, Spanish, Italian, and Dutch (Rangel Pardo et al., 2015).

Only a few studies have attempted personality trait inference from Russian social media text. Stankevich et al. (2018) learned a three-way (low, medium, high) classification of the five factors from a dataset of 165 VKontakte profiles. However, due to the sparseness of usable, user-generated text in their dataset, they were unable to use lexical features, but only very basic features on the text (such as average numbers of words and sentences, use of punctuation and uppercase). Their reported F1 scores range from 36% for Conscientiousness to 53% for Agreeableness.

Similarly, Ignatiev et al. (2019) used both SVM and Random Forest approaches for a two-way (highest quartile, lowest quartile) classification of five factors traits from their dataset of 1,020 VKontakte profiles. They used lexical features, an aggression lexicon, user profile information, and a repost matrix, and reported F1 scores ranging from 61.75% on Openness to Experience to 73.75% on Extroversion.

Litvinova et al. (2015; 2016) and Vybornova et al. (2011) infer personality traits from other genres of Russian text (besides social media such as VKontakte). They propose several and test linguistic correlates to personality, such as content/function words, readability indices, lexical diversity and usage of first-person singular pronouns.

3 RFET Features

3.1 Morphological

Morphological features are generated using PyMorphy2, a Russian morphological analyzer and inflection engine (Korobov, 2015).\(^4\) The morphological analyzer is able to detect the following morphological features: part of speech, animacy, verbal aspect, dictionary citation forms, case, gender, involvement, mood, number, person, tense, transitivity and voice. Because PyMorphy2 evaluates on the token level, without looking at the context, syntax level features are not captured by the toolkit.

The toolkit leverages the output of PyMorphy2 to calculate the frequencies of these features within the text, but we added extensions to produce additional linguistic features for RFET. We diverge from PyMorphy2 where the tool conflates logically different phenomena. For example, PyMorphy2 determines gender of nouns and adjectives by word ending; however, some borrowed nouns and acronyms are indeclinable and thus may not show any cues for gender. Not all such nouns are neutral, so assigning a default gender risks inaccuracies (Wang, 2014). Inevitably, some lexical items will be missing from PyMorphy2’s lexicons and thus have undetermined gender so far as PyMorphy2 is concerned.

Additionally, not all Russian parts of speech inflect for gender. The toolkit expands the tag set from 4 tags (masculine, feminine, neuter, null) to 5 tags (masculine, feminine, neuter, undetermined, non-gendered-pos) to differentiate between those words of specific parts of speech that are not gendered in Russian (i.e. conjunctions, comparatives, gerunds, adverbs, particle, infinitives, prepositions and predicatives) with those parts of speech that can have gender, but are not determined by PyMorphy2.

RFET also utilizes PyMorphy2’s OpenCorpora Dictionary (Открытый корпус), to identify the ratio of words in the text that are not found in its lexicon. This ratio may indicate usage of new words, slang, typos, URLs and other Out Of Vocabulary (OOV) items. The ratio of OOV items can be a useful feature for classification to sociolinguistic targets.

3.2 Laughter, Emoticons and Emoji

While laughter appears in informal text across languages, the characters and patterns used

\(^4\)https://github.com/kmike/pymorphy2. At the time of RFET’s design, PyMorphy2 seemed to be the most widely cited specifically Russian morphological parser and was integrated into DeepPavlov. UDpipe’s Russian lemmatizer may have been a reasonable alternate choice. A Russian lemmatizer for SpaCy was not available until February 2021, as documented in the SpaCy blog. (https://explosion.ai/blog/spacy-v3). A comparison of lemmatizers is out of scope for this paper.
to represent laughter differ. RFET returns frequency features on Russian-specific laughter and emoticons found on social media, as well as language-agnostic emoji features. These features could be utilized in an author attribution system, sentiment analysis, or emotion classification system.

RFET returns frequency information on type, number of times used and length of laughter or emoticon within the text. RFET tracks the following types of laughter (in featurename (example) pairs): haa (xaha), haah (xahaaxaax), haahaa (xahaaxa), ha-ha (xa-xa), hehe (hehhehee), hihi (xihxihxhi), hi-hi (xih-xih), Lol (xol), Lolol (xolol), phaha (pahaaxa), HAHAXA-RAHAH, hoho (xhoxo), HIHI (XIXHI), HEEHEE (XEXEXE), HOHO (XOXO), HA-HA (XA-XA), HIIHI (XHI-XHI), as well as Russian happy face parenthesis (“))))
5 and sad face parenthesis (“((“)). These are implemented as templated regular expressions that match and report the length of variants.

RFET also has a feature which returns the frequency of emoji usage and leverages Cal Henderson’s emoji-data package.6

3.3 Sentiment and Emotion Features

Sentiment analysis is one of the most popular commercial applications of NLP, and RFET makes it easier for a nonnative speaker to implement a system in Russian. Currently RFET utilizes one emotion/sentiment lexicon and can easily be extended to allow for others.

The NRC Emotion Lexicon (Mohammad and Turney, 2013), also known as EmoLex, is a lexicon translated into 104 languages. Each lexical entry is coded for Positive and Negative sentiment and the emotions Fear, Anger, Sadness, Joy, Disgust, Surprise, Trust and Anticipation.7 RFET reports the emotions present in the text by using PyMorphy2 to resolve the dictionary citation form of each token in the text and reporting the emotion(s) and sentiment orientation associated with that form in the lexicon, if available. If the token is present in the lexicon but is not coded as positive or negative, the token is coded as neutral. Similarly, RFET reports whether a token is not found in the lexicon. RFET uses these resources to return a dictionary of sentiment (positive, negative, neutral) and emotion (fear, sadness, joy, disgust, surprise, trust, and anticipation) or “token_not_in_lexicon” and the count of tokens representing these.

3.4 Lexical Diversity

According to Litvinova et al. (2016, 2017), a lack of lexical diversity was associated with individuals with a greater likelihood of self-destructive behavior, which may be useful to author profiling or personality identification systems.

Litvinova et al. (2017) described lexical diversity through a variety of features, including type to token ratio, an index of formality, an index of lexical density, the ratio of function words (particles, prepositions, conjunctions, etc) to total tokens, the ratio of content words (nouns, verbs, infinitives, adjectives, adverbs, etc.) to total tokens, the ratio of personal pronouns to total tokens, and the proportions of the 100 most frequent Russian words in the document to all tokens.

RFET implements these key lexical diversity features replicating Litvinova’s descriptions; in addition to extracting the proportion of the top 100 most frequent Russian words (unigrams), it also tracks usage of the top most frequent bigrams, trigrams, and 4-, 5-, and 6-grams in the Russian National Corpus (RNC).8

3.5 Other Lexical Features

Other Russian specific features that RFET extracts and quantifies are punctuation, digits, diacritics, other languages, and other scripts. The punctuation, diacritics, and quotation scripts include Russian specific unicode characters (i.e. «, », „ and “) in addition to the punctuation that is shared across languages. The punctuation feature does overlap with the emoticon features.


7https://github.com/iamcal/emoji-data


N-grams are found at e.g., https://ruscorpora.ru/old/1grams.top.html.
The ratio of code switching or borrowings from Western languages may be indicated by the ratio of Latin characters to total characters in the text, and this is reported by RFET as an additional feature. Another feature utilizes a language identification package\(^9\) to determine the language of the text, in order to identify instances of non-Russian text within a corpus of presumed Russian documents, including languages such as Bulgarian, Macedonian, and Ukrainian that also use Cyrillic.

4 Inferring Personality Traits

4.1 Dataset

The dataset for evaluating the personality trait inference task was collected by the authors during 2020. It consists of 149K VKontakte posts from 288 consenting participants, with a total of 3.8M word tokens. (This corpus was filtered from a larger collection by excluding posts containing URLs, ASCII art, duplicate posts, and posts that appeared to be auto-generated. It includes only those participants with at least 1200 tokens in their VK posts after this filtering process.) Each of the 288 participants took the Russian version of the BFI-2 inventory (Soto and John, 2017; Shchebetenko et al., 2020). The labels (personality trait scores) were rescaled from the raw inventory scores to the interval \([-5, 5]\). The dataset was partitioned by author into train and test sets (with 80% of author accounts comprising 79% of the total word tokens in train and the rest in test).

4.2 Baselines

We created two sets of baseline models for personality ID, one using classical machine learning methods, and one using neural embeddings.

The first set of baseline models created for the five personality traits used a standard bag-of-words approach using term frequency–inverse document frequency (tf*idf) and included language-independent features of “lexical richness” such as Yule’s (2014) \(K\), Sicil’s (1975) \(S\), Honoré’s (1979) Measure, Brunet’s (1978) Measure, Maas’s (1972) \(a^2\), and Rubet’s \(k\) (Dugast, 1979). None of these features require specific knowledge of the target language. Because of the size of the \(tf*idf\) data, principal component analysis (Jolliffe and Cadima, 2016) was used to reduce the number of features. In total, 73 features were kept from the \(tf*idf\) data, which corresponded to keeping 85% of the total variance, as well as the six lexical richness features.

The second set of baseline models created for the five personality traits used Sentence-BERT (SBERT) (Reimers and Gurevych, 2019) in combination with Russian-specific language model Sentence RuBERT (rubert-base-cased-sentence)\(^{10}\), a fine-tuned version of RuBERT (Kuratov and Arkhipov, 2019) for sentence embedding. Sentence-BERT produces fixed-sized embeddings by pooling the output of a language model. These fixed-sized embeddings are convenient to use in combination with classification algorithms for sentence classification tasks. We used Sentence-BERT’s highest performing pooling strategy to produce these embeddings by taking the mean of all output language vectors and Sentence-BERT’s default maximum sequence length of 128 WordPieces.\(^{11}\)

4.3 Method

We used a standard implementation of a support vector machine (SVM) to train personality identification regression models from a corpus of Russian VKontakte text labeled for Big Five personality traits, varying the features supplied to each system. Each feature set we utilized was used to train and test against the five traits-Openness (O), Conscientiousness (C), Extraversion (E), Agreeableness (A), and Negative Emotionality (N) traits. A linear kernel was used for all traits, and the regularization parameter \(C\) was tuned separately for each trait.

The target data are scaled to the interval \([-5, 5]\) for each trait, and we report the root

\(^9\)https://pypi.org/project/langdetect/, which is a Python port of Shuyo Nakatani’s (2010-2014) language-detection Java language ID package.

\(^{10}\)https://huggingface.co/DeepPavlov/rubert-base-cased-sentence

\(^{11}\)Preliminary experiments suggested that using a maximum sequence length of 512 WordPieces did not meaningfully improve performance accuracy, while adding significant time requirements. The average post in our training corpus, when tokenized to the rubert-base-cased-sentence lexicon, was 50 WordPieces. There were 4719 posts longer than 128 WordPieces, about 4.5% of our training corpus, which means with a maxlength of 128, only 4.5% of our samples were clipped.
mean square error (RMSE). Across all experiments, the same training and testing splits were used on the data.

Our goal is to predict all five personality traits, so this problem can be treated as a multi-target regression problem. Since there is evidence in our dataset and other literature that personality traits are correlated (Gosling et al., 2003), the target trait of one model may be useful as a feature when predicting another trait. Because of this, predicted values for some traits were fed back in as features for other models, called “stacked single target” (SST) chains (Spyromitros-Xioufis et al., 2016). Similar work has been done for other multi-target regression problems with positive results (Melki et al., 2017). It is possible for this process to be optimized, standardized, or generally improved. We look to investigate this in future work. For the experiments reported in Table 1, the trait that was predicted the most accurately was the one chosen as a feature for the next round of models. Each trait was added in as a feature exactly one time, such that the final database consisted of all the raw data, plus five new features (one for each trait). Thus each of the five traits is predicted using the stacked single targets (i.e., trait value predictions from previous iterations of the model) for the four other traits as features.

4.4 Results

Table 1 shows root mean squared error (RMSE) (lower is better) and \( R^2 \) (higher is better) for our bag of words classifier with and without the RFET features. Each post was treated as a unique entry, and the data was fed into a support vector regression model. To prevent overfitting, both the test-train split and cross validation splits were done on the participant level. The model never saw any data from any participants in test or validation sets. The inclusion of RFET features produced a more accurate model based on both RMSE and \( R^2 \) values. We believe these features add significant value to modeling personality traits and likely other tasks as well.

Our Sentence-BERT neural baseline results for post-level predictions of a post author’s personality traits, as well as comparable predictions for the BoW+RFET model, can be found in Table 2. Unlike our SVM baseline and SVM with RFET features results in Table 1, neither Sentence-BERT nor the SVM with RFET features shown here utilize SST as input to help improve the performance. We found that the SVM with SST and SVM with RFET features (with or without SST) outperformed the Sentence-BERT neural baseline.

Comparing the two BoW+RFET columns in Tables 1 and 2, we can see that the SST chains do improve the BoW+RFET model’s accuracy for all features but Negative Emotionality, but of course this comes at the cost of speed, as the five traits can no longer be trained or decoded in parallel (and training requires several more iterations).

5 Discussion

For the personality trait identification task on this social media dataset, we see that a classic SVM baseline using RFET features outperforms a similar SVM without the RFET features. We likewise see that the SVM with RFET features outperforms a model using a transformer model pre-trained on Russian text.
One trait where the advantage of the RFET features is particularly large is the Agreeableness trait. One possible explanation for this may be differences in participants’ use of emoji, emoticons, and/or emotional words correlating with their Agreeableness trait values. RFET includes features specifically developed for emoji, emoticons, and emotional words, and even the SVM BoW model may be somewhat sensitive to them; the pretrained model, on the other hand, may be ignoring emoji and emoticons, since such “words” may not have appeared in its original training data.

Another advantage the standard machine learning methods have over neural models is interpretability. Since a SVM was used with a linear kernel, it is possible to extract feature importance from each model, and gain insights into where the strongest correlations lie. To do this, we employed the R package e1071 (Meyer et al., 2021). Every model is trained independently, and so produces different feature weights, but in general, these features appeared as the most important for our machine learning models:

- content, function word to token ratios;
- NRC emotional lexicon tokens;
- number frequency (singular vs. plural)
- frequency of morphological features: grammatical number, animacy, case, verbal mood
- frequencies of top 100 (RNC) unigrams

6 Limitations and Future Work

Neural NLP models that follow the BERT architecture, like Sentence RuBERT, grow in memory requirements quadratic to the sequence length. Because of this, the models are limited to a sequence length (often 512 WordPieces, but 128 here) with the remaining WordPieces in a post ignored and left unprocessed. VK posts longer than 128 WordPieces are clipped in our Sentence-BERT experiments, while the full posts serve as input to the SVM bag of words and RFET feature extraction systems. Model architectures have been released catering to long form English text, such as Big Bird (Zaheer et al., 2020) and Longformer (Beltagy et al., 2020), but models using these architectures are not yet available for Russian.

Lastly, RuBERT was trained on Wikipedia and books. Our experiments were on social media text, which RFET was designed to address. This genre mismatch (and RuBERT’s resulting limited coverage for WordPieces specific to social media) may have limited Sentence RuBERT’s effectiveness for this task.

One possible weakness in the RFET model (as trained on these training data) is sparsity of certain features. Certain RFET features may be overfitting to particular participants. Idiosyncratic uses of words from a few people with unusual personality trait values or a large volume of posts may be inappropriately generalized as signals for those trait values.

One possible mitigation, of course, is the collection of data from a much larger set of individuals. In these experiments, we apply a more local mitigation strategy: combining different kinds of laughter into a single feature. One strength of RFET is the flexibility of combining or splitting apart features according to the need of the task. For example, personality trait (or other attribute) detection may benefit from combining features; author identification or verification may benefit from keeping very specific features split apart.

Future feature extractors include frequency of Russian diminutives through the usage of suffixes and infixes, ratio of other language scripts being used in the text and usage and frequency of filler words and phrases. Additional information about sentiment and prevalence of emojis will be incorporated from the Emoji Sentiment Ranking (Kralj Novak et al., 2015), which provides usage statistics and associated sentiment for each emoji. An expanded list of emoticons will also be added to reflect the frequency and usage of more emoticons.

Future iterations of RFET may utilize RuSentLex (Loukachevitch and Levchik, 2016), a Russian emotion lexicon with 16,057 entries. Each entry includes a description of its syntactic category, a lemmatized version, the sentiment valence, and source of the valence (opinion, feeling, or fact). Ambiguous entries in RuSentLex (with more than one possible value for valence or source) are also elaborated with examples. We anticipate these features will provide better coverage of sentiment than the

12http://kt.ijs.si/data/Emoji_sentiment_ranking/
7 Conclusion

This paper introduces the Russian Feature Extraction Toolkit, an API for feature extraction on the Russian language. Each feature in the toolkit utilizes linguistic knowledge of the Russian language. It is designed to get a Russian non-speaker up and running quickly on Russian NLP tasks, and to speed up the workflow of Russian speaking NLP programmers. We have shown that it improves performance on a Big Five personality trait inference task relative to a SVM baseline with only language-independent features and, more surprisingly, to a pre-trained transformer baseline using Sentence RuBERT. This suggests that RFET's features specific to social media can be very useful for enhancing state-of-the-art methods for certain genres or domains.

Licensing: We plan to release RFET for non-commercial research and education. A public API will be made available for demonstration purposes. For commercial licenses, contact the University of Maryland’s Office of Technology Commercialization.  

Ethical Considerations

No novel data collection was done specifically for developing RFET; RFET features depend on pre-collected corpora such as the Russian National Corpus (RNC). Data collection for the evaluation of RFET’s efficacy for personality trait estimation (as described in section 4.1) was conducted with the approval of the University of Maryland Institutional Review Board (IRB). During the consent process, potential respondents were informed of the research purposes, and that the researchers would remove, anonymize, or pseudonymize names of entities in the collected social media data deemed to risk personally identifying the participant prior to any sharing of the data with those outside the IRB protocol. Any additional potential risks to confidentiality have been minimized by keeping all data on a secure Amazon Web Service (AWS) server, to which only authorized researchers with the requisite IRB training have access. Non-anonymized data will be destroyed upon the close of the IRB protocol; only data which has undergone our de-identification process would be retained.

Although we set collection targets by gender and age bracket to obtain representative samples to encourage greater equity of representation by gender and age, the use case evaluation was naturally biased towards individuals who write and post lots of text on social media (for which women and younger writers were over-represented in our sample). An evaluation which used equal amounts of text for each individual would avoid this bias, at the cost of leaving unused large portions of the corpus.

Since RFET is a toolkit to assist researchers in improving their own NLP applications, the primary beneficiaries are NLP researchers and developers, particularly those working with social media. Likewise, the main source of potential harm lies with what researchers and developers decide to do with the RFET tool. (Personality trait inference is just one example of potential downstream applications and its own ethics of use depend on where and for what purpose it is applied.) Biases may exist in the older texts used (e.g., some of those in the RNC) but since the features used here are largely based on grammatical categories and lists of keywords, the toolkit is arguably more transparent than tools based on semantic embeddings and such bias easier to identify and address.

Acknowledgments

The authors thank Nadiya Klymenko and Ali Bhatti for work on early versions of RFET, Ewa Golonka for Russian language expertise, Jarrett Lee for technical support, and Mike Bunting, Laurel Miller-Sims, Tom Conners, Michelle Morrison, Aric Bills, and several others for assistance in the data collection. We also thank anonymous reviewers whose reviews improved the presentation of the work.

References


Abstract

In this study, we proposed a novel Lexicon-based pseudo-labeling method utilizing explainable AI (XAI) approach. Existing approach have a fundamental limitation in their robustness because poor classifier leads to inaccurate soft-labeling, and it lead to poor classifier repetitively. Meanwhile, we generate the lexicon consists of sentiment word based on the explainability score. Then we calculate the confidence of unlabeled data with lexicon and add them into labeled dataset for the robust pseudo-labeling approach. Our proposed method has three contributions. First, the proposed methodology automatically generates a lexicon based on XAI and performs independent pseudo-labeling, thereby guaranteeing higher performance and robustness compared to the existing one. Second, since lexicon-based pseudo-labeling is performed without re-learning in most of models, time efficiency is considerably increased, and third, the generated high-quality lexicon can be available for sentiment analysis of data from similar domains. The effectiveness and efficiency of our proposed method were verified through quantitative comparison with the existing pseudo-labeling method and qualitative review of the generated lexicon.

1 Introduction

Sentiment analysis is employed to identify the sentiment orientation and measure the emotional strength (Khan et al., 2016; Khan & Lee, 2019; Silva et al., 2016). To better understand information generated by online user and to take advantage of it, sentiment analysis is becoming major topic in text mining field in last two decades (Duan et al., 2020; Nagarajan & Gandhi, 2019; Valdivia et al., 2017). Previous studies of sentiment analysis can be roughly categorized into two different groups: 1) lexicon-based approaches and 2) machine learning-based approaches (Khan et al., 2019; Khoo & Johnkhan, 2018).

The lexicon-based approaches efficiently calculate the sentiment score of sentence of document since it does not need to train the classification model in advance. However, The lexicon-based approaches depends on the availability of a sentiment lexicon which is collection of manually pre-created sentiment words lexicon and its sentiment polarity (Huang et al., 2020; Taj et al., 2019; Alqaryouti et al., 2019).

Meanwhile, the machine-learning based approaches require a training set consists of labeled data (e.g. positive, negative or neutral). To address the challenge caused by limited labeled data, which is the usual case in practice, semi-supervised learning have attracted more attention recently (Han et al., 2020; Zhang et al., 2020; Lee et al., 2019). The semi-supervised learning can be divided into several categories such as consistency regularization approaches, entropy minimization approaches and augmentation based approaches and pseudo-labeling approaches.

Among those approaches, the pseudo-labeling approaches are one of the most intuitive and widely used semi-supervised learning in sentiment analysis (Xu & Tan, 2019; Wu et al., 2019; Chen et al., 2020). The pseudo-labeling approaches tried to train the sentiment classification model with small number of labeled data and add unlabeled data with high-confidence of sentiment score, calculated by
trained model, into the labeled dataset in each learning cycle. However, existing approaches have a fundamental limitation in their robustness because soft labeling task and classification task completely depends on each other. That is, poor classifier leads to inaccurate soft-labeling, and it lead to poor classifier repetitively (Van Engelen & Hoos, 2020; Devgan et al., 2020). The left illustration in Figure 1 illustrate the limitation of existing approaches of pseudo-labeling. Thus, this study proposes the robust pseudo-labeling approaches by combining heterogeneous frameworks of sentiment analysis. And the performance of the proposed method will be justified by comparing the two changes in accuracy with graphs.

2 Literature Review

2.1 Studies on semi-supervised learning in sentiment analysis

As a fore-mentioned, the semi-supervised learning can be divided into several categories such as consistency regularization approaches, entropy minimization approaches, augmentation based approaches and pseudo-labeling approaches. The principle of consistency regularization underlines that the model predictions should be less sensitive to the extra perturbation imposed on the input samples (Yu et al., 2020). The entropy minimization approaches encourage the model to output confident predictions on unlabeled data, and the augmentation based approaches are methods of generating various augmented data and using it for learning (Tu & Yang, 2019).

Among those approaches, the pseudo-labeling approaches such as self-training (pseudo-labeling) or co-training is one of the most intuitive and widely used semi-supervised learning in sentiment analysis. In self-training, the most confident unlabeled data with their predicted label, are selected to add to the training set. (Baugh, 2013) employ the self-training for increasing the size of the feature space and (Becker et al., 2013) adapt a static polarity lexicon along with self-training to increase the number of labeled dataset. (Haimovitch et al., 2012) makes use of self-training for large-scale reviews of polarity prediction and (Wang et al., 2016) apply the self-training into text sentiment classification to improve the quality of the training text. (Hajmo-hammadi et al., 2016) utilized semi-supervised self-training approaches to incorporate unlabelled sentiment documents from the target language in order to improve the performance of cross-lingual methods.

And co-training assumed that feature space can be divided into two different views. Two different classifiers are trained with the labeled data, and then applied to the unlabeled data to add them into trained set with confidence level of prediction. (Yu et al., 2014) focuses on revisiting co-training in depth and discusses several co-training strategies for sentiment analysis following a loose assumption and (Zhang et al., 2014) applies co-
training to select the most reliable instances according to the two criteria of high confidence and nearest neighbor for boosting the classifier, also exploit the most informative instances with human annotation for improve the classification performance. (Wang et al., 2014) implemented co-training on multiple component learner of different types to allow performance of their respective advantages and (Xia et al., 2015) propose a dual-view co-training algorithm based on dual-view document representation for semi-supervised sentiment classification. (Catal & Nangir, 2017) investigate the potential benefit of multiple classifier systems concept on Turkish sentiment classification problem with co-training approach and (Li et al., 2019) proposed semi-supervised learning approach based on the hybrid mechanism of self-learning for textual sentiment classification.

However, the high confidence is not necessarily correct with aforementioned approaches. Label error will be transferred and accumulated in the training and labeling process, and it lead the unstable semi-supervised learning process without robustness. Thus, we proposed robust semi-supervised learning approach by combining heterogeneous framework based on auto-generated lexicon.

2.2 Studies on explainable model

In order to generate the lexicon utilized for soft-labeling automatically. We utilized numerous explainable models such a Local Interpretable Model-Agnostic explanations (LIME), SHapley Additive exPlanations (SHAP), Layer-wise Relevance Propagation (LRP) and Gradient-weighted Class Activation Mapping (Grad-CAM) respectively or in and ensemble. And, we also utilize linear model-agnostic XAI methods such as Logistic regression (LR) and support vector machine (SVM) in our experiments.

The key intuition behind LIME is that it is much easier to approximate a black-box model by a simple model locally (in the neighborhood of the prediction we want to explain), as opposed to trying to approximate a model globally. This is done by weighting the perturbed images by their similarity to the instance we want to explain (Hu et al., 2018; Lee et al., 2020).

SHAP is a method to explain individual predictions. SHAP is based on the game theoretically optimal Shapley Values. SHAP values for each feature represent the change in the expected model prediction when conditioning on that feature. For each feature, SHAP value explains the contribution to explain the difference between the average model prediction and the actual
prediction of the instance (Adadi & Berrada, 2018; Lundberg & Lee, 2017).

Grad-CAM uses the gradients of any target prediction flowing into the certain convolutional layer in CNN model to produce a coarse localization map highlighting the important regions in the image for predicting the class of the image (Lee et al., 2020; Selvaraju et al., 2017). We modified the Grad-GAM algorithm to be applied for textual sentiment classification.

LRP is a method to compute scores for image pixels and image regions denoting the impact of the particular image region on the prediction of the classifier for one particular test image (Binder et al., 2016). We also modified the LRP algorithm to be applied for textual sentiment classification.

3 Method

As aforementioned, instead of existing approaches which calculate the confidence of sentiment score for unlabeled dataset by same classifier, we calculated the confidence of sentiment score for unlabeled dataset by auto-generated lexicon. That is, a lexicon is automatically generated through the explainability score calculated while learning the classifier with labeled data, and pseudo-labels are assigned to the unlabeled dataset based on the generated lexicon. The summary of our proposed method is illustrated in Figure 2.

In detail, the learning process in the work of creating lexicon is as follows. First, a binary classification model is trained using data with positive and negative labels. And the importance score of each word is grasped through the coefficients derived from each model. An initial lexicon is created based on this importance score.

Second, pseudo-labels are additionally assigned to N unlabeled data using the generated lexicon, and the previous process is repeated using N additional data, and the lexicon is updated based on this result.

When creating a lexicon, each word's importance score is assigned as the average of the word's scores each time the dictionary is updated.

By repeating the process, the lexicon is updated and the process of assigning pseudo-labels to unlabeled data is completed. These processes are defined formally in the Algorithm 1.

Algorithm 1 Creating Sentiment Lexicon

1: Obtain a small set of L of labeled examples
2: Obtain a large set of U of unlabeled examples
3: for N iterations do
4:   for each explainable classifier Ci do
5:      Learn classifier Ci from L
6:   end for
7:   Update lexicon D from ensemble of Ci
8:   Choose confidently predicted example E from U based on normalized D
9:   E is removed from U and added (with their given labels) to L
10: end for

3.1 Details for creating lexicon

As mentioned in the previous section, several criteria were used in updating the lexicon in the proposed method. This section describes the details applied to update the lexicon. And the basic parameter settings used in each methodology are defined in the Table 1. The setting of each parameter was selected based on experience in various experiments.

<table>
<thead>
<tr>
<th>Explainable Method</th>
<th>Embedding Method</th>
<th>Hyper Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>TF-IDF</td>
<td>λ = 0.1, δ = 0.2</td>
<td>-</td>
</tr>
<tr>
<td>LIME</td>
<td>TF-IDF</td>
<td>α = 20, β = 10, γ &gt; 0.8</td>
<td></td>
</tr>
<tr>
<td>SHAP</td>
<td>TF-IDF</td>
<td>α = 20, β = 10, γ &gt; 0.8</td>
<td></td>
</tr>
<tr>
<td>Grad-CAM</td>
<td>Word2vec</td>
<td></td>
<td>θ &gt; 0.25 or θ &lt; 0.75</td>
</tr>
<tr>
<td>LRP</td>
<td>Word2vec</td>
<td></td>
<td>θ &gt; 0.25 or θ &lt; 0.75</td>
</tr>
</tbody>
</table>

Table 1: Entire parameter setting

3.1.1 Linear model-agnostic approaches

LR and SVM calculate the importance of words using the coefficient values of the classification model. The two models vectorized data and trained the model using Term Frequency – Inverse Document Frequency (TF-IDF).

Calculate the importance score of each word and build a dictionary using only words with a score of λ (= 0.1) or higher. And in the process of updating dictionaries, only words with an importance score of δ (=0.2) or higher are used.

When using a Support Vector Machine, a lexicon was created through a binary classifier through Support Vector Classifier. Similar to Logistic Regression, words were generated by calculating
regression coefficients for each word learned in the model.

3.1.2 XAI-based approaches

Unlike linear model agnostic approaches, in XAI-based approaches, scores are assigned to words per sentence. That is, in the process of constructing a lexicon, a score is calculated and updated one by one.

In this study, the importance of words was calculated for each sentence by applying LIME and SHAP to the model trained by Logistic Regression, and the importance of words was calculated for each sentence through Grad-CAM for the model trained with CNN and LRP for the model trained with LSTM. We proceeded to calculate the importance. LIME and SHAP used TF-IDF matrix to vectorize sentences, and CNN and LSTM training data used Word2Vec to embed sentences.

To construct a meaningful lexicon, not all sentences are used for lexicon construction, but only sentences with a predicted value of 0.8 or higher are used, and only the top 20 words of importance score are used in each sentence. In the process of updating a dictionary, the dictionary is updated using only the top 10 words in the sentence.

The process of building a lexicon through Grad-CAM and LRP, use only sentences with sigmoid values greater than 0.75, or less than 0.25, close to zero and one.

Lastly, considering the characteristics of Grad-CAM, which does not show directionality, the frequency of each word in the positive lexicon and the negative lexicon is compared and set as a positive word or negative word.

4 Experiment

4.1 Data description

In this study, experiments were conducted using 7 open datasets. The data used were composed of various domains such as movies, accommodations, games, shopping, airlines, and clothing. The Table 2 below summarizes the description of the dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Num. of instance</th>
<th>Pos.</th>
<th>Neg.</th>
<th>Maximum length of reviews</th>
<th>Average length of reviews</th>
<th>Num. of vocabs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airline review</td>
<td>74,623</td>
<td>37,352</td>
<td>37,271</td>
<td>385</td>
<td>23</td>
<td>1,146</td>
</tr>
<tr>
<td>Amazon review</td>
<td>400,000</td>
<td>200,000</td>
<td>200,000</td>
<td>86</td>
<td>30</td>
<td>1,887</td>
</tr>
<tr>
<td>Clothing review</td>
<td>23,486</td>
<td>19,314</td>
<td>4,172</td>
<td>54</td>
<td>24</td>
<td>1,046</td>
</tr>
<tr>
<td>Hotel review</td>
<td>38,932</td>
<td>26,521</td>
<td>12,411</td>
<td>606</td>
<td>71</td>
<td>2,301</td>
</tr>
<tr>
<td>IMDB review</td>
<td>25,000</td>
<td>12,500</td>
<td>12,500</td>
<td>776</td>
<td>101</td>
<td>4,366</td>
</tr>
<tr>
<td>Steam review</td>
<td>17,494</td>
<td>9,968</td>
<td>7,526</td>
<td>900</td>
<td>64</td>
<td>2,213</td>
</tr>
<tr>
<td>Yelp review</td>
<td>38,000</td>
<td>19,000</td>
<td>19,000</td>
<td>381</td>
<td>56</td>
<td>2,451</td>
</tr>
</tbody>
</table>

Table 2 : Summary for dataset

4.2 Experiment setup

In the experiment, we basically verify that the proposed method shows higher performance and robustness than the existing pseudo-labeling method in each dataset. For the performance comparison in the same experimental setting, the same baseline architecture was used, and accordingly, a one-to-one comparison was performed as follows: 1) LR based existing pseudo-labeling approach vs. LR based proposed method, 2) SVM based existing pseudo-labeling approach vs. SVM based proposed method, 3) LR based existing pseudo-labeling approach vs. LIME based proposed method, 4) LR based existing pseudo-labeling approach vs. SHAP based proposed method, 5) CNN based existing pseudo-labeling approach vs. Grad-CAM based proposed method, and 6) LSTM based existing pseudo-labeling approach vs. LRP based proposed method.

The experimental setup of the proposed method is as follows. First, the initial emotion lexicon is constructed using 1000 positive and 1000 negative sentences. Then, based on the lexicon, pseudo-labeling is repeatedly performed by 1000 pieces, and the emotional lexicon update is performed again using the data. In addition, 1000 pieces that were not used for learning were set as test data, and the change in accuracy of pseudo-labeling based on the lexicon was measured.

The experiment of the existing pseudo-labeling methodology was carried out as follows. As in the experiment of the proposed methodology, a classifier is created using 2000 data (1000 positive sentences and 1000 negative sentences), and prediction is performed with an additional 1000 data units. Among them, pseudo-labeling was performed on 100 data, which are the top 10% of
the predicted values, and the change in accuracy of this data was measured.

4.3 Experiment result

Throughout the model, the figures of accuracy for the data were shown similarly. Initially, the accuracy of the automatic generation lexicon was lower compared to the general semi-supervised learning method, but over time, the accuracy of the semi-supervised learning decreased and the accuracy of the proposed method was maintained or increased.

In this study, we conducted an experiment comparing the accuracy of the automatic generation-based method using six methods and the existing method using seven data. In this section, we present Figure 3, only graphs comparing the accuracy of methods that conducted Pseudo-labeling based on CNNs and the accuracy of proposed methods that utilize automatic generative lexicons based on Grad-CAM. In the case of Grad-CAM, it can be seen that the proposed method shows better performance than the existing method at all times. In particular, in the case of the existing methodology, it can be confirmed that the classification performance rapidly decreases after the initial classification performance is poor.

5 Conclusion

In this study, a novel Lexicon-based pseudo-labeling method utilizing XAI approach was proposed that improved the limitations of the existing pseudo-labeling method. The existing approaches have a fundamental limitation in their robustness because soft labeling task and classification task completely depends on each other.

However, the proposed methodology automatically generates a lexicon based on XAI and performs independent pseudo-labeling, thereby guaranteeing higher performance and robustness compared to the existing one. In addition to robustness, since dictionary-based pseudo-labeling is performed without re-learning, time efficiency is considerably increased, the generated high-quality lexicon can be available for sentiment analysis of data from similar domains.

The quantitative excellence of the proposed method was verified through a one-to-one performance comparison with the existing method, and the effectiveness and efficiency of the proposed method were qualitatively verified by reviewing the generated lexicon.

Future research may extend the scope of XAI based lexicon construction in a more general point of view. As shown in the experimental results, there are differences in lexicons for each domain, and a study to construct a general-domain lexicon by integrating them is presented as a future work. Moreover, such studies can be expected to aid in the widespread application of the proposed semi-supervised learning in various tasks arising within the natural language processing domain.

Acknowledgments

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2020R1F1A1067914).

References


Multiple Teacher Distillation for Robust and Greener Models

Artur Ilichev
Moscow Institute of Physics and Technology, Moscow, Russia
ilichev.as@phystech.edu

Nikita Sorokin
Huawei Noah’s Ark lab, Moscow, Russia
lastname.firstname@huawei.com

Valentin Malykh

Irina Piontkovskaya

Abstract

The language models nowadays are in the center of natural language processing progress. These models are mostly of significant size. There are successful attempts to reduce them, but at least some of these attempts rely on randomness. We propose a novel distillation procedure leveraging on multiple teachers usage which alleviates random seed dependency and makes the models more robust. We show that this procedure applied to TinyBERT and DistilBERT models improves their worst case results up to 2% while keeping almost the same best-case ones. The latter fact keeps true with a constraint on computational time, which is important to lessen the carbon footprint. In addition, we present the results of an application of the proposed procedure to a computer vision model ResNet, which shows that the statement keeps true in a totally different domain.

1 Introduction

Nowadays the language models became a cornerstone in many natural language processing tasks. Their results in the benchmarks show new high scores. But with great power sometimes comes huge size, the current models could have dozens of billions of weights, e.g. TuringNLG (Rasley et al., 2020), to GPT-3 (Brown et al., 2020) with 175 billion parameters, and counting. In many cases the resources of computational memory are limited and there is a demand for small solutions. One of such solutions is a distillation of language models. There were presented several approaches for the specified task, among others these are TinyBERT (Jiao et al., 2019) and DistilBERT (Sanh et al., 2019).

We analyzed these approaches and found that they share an important flaw - the dependency from the random seed used in the distillation process.

During the distillation a student model needs to be trained multiple times with different random seeds to achieve better performance, although it is not guaranteed that there will be “winning numbers” in your seed choice. So we concentrated on the worst case scenario and proposed a technique to improve it. Considering the computational resources, the improvement could be achieved with the same computational budget, allowing one to diminish the carbon footprint. We propose the novel technique of multi-teacher distillation called to make the mentioned language models more robust to seed selection. We evaluated our method on a computer vision classification model ResNet (He et al., 2016) and make sure that the proposed technique is applicable to a totally different domain.

Our contribution is as follows: we present (i) a new distillation method and an experimental evaluation of this method for three models, namely (ii) TinyBERT and (iii) DistilBERT, where we modified the distillation procedure adding the task-specific distillation, for three natural language understanding tasks and (iv) ResNet for a computer vision task, showing on the one hand that models learned from multiple teachers are consistently better in the worst case and about the same in the best case, and on the other hand, these models are better with a constraint on computational time.

This work is structured as follows: in Section 3 we describe the distillation process and our modification (in Section 3.5); in Section 4 we describe the datasets used in the experiments, which are described in Section 5. The Section 6 concludes the article and discusses the obtained results.

2 Related Work

Common techniques for model compression and acceleration can be roughly grouped into three groups. Pruning parts of large-scale models allows to re-
duce the number of weights and accelerate inference. Sajjad et al. (2020) drop entire layers from pre-trained Transformer models, showing that several top-layers can be dropped, maintaining the performance on downstream tasks. Michel et al. (2019) remove all but one attention head, showing that indeed one head might be sufficient at the test time not only for sentence modeling tasks and but for machine translation also. Quantization keeps the network structure unchanged, but quantizes network weights to smaller data types, such as int8. Quantization can be performed both post training (Bhandare et al., 2019) or during fine-tuning (Zafrir et al., 2019). Knowledge distillation (Hinton et al., 2014) trains the more compact models, students, to reproduce the behavior of a larger model, the teacher. BERT-PKD (Sun et al., 2019), TinyBERT (Jiao et al., 2019) and DistilBERT (Sanh et al., 2019), distilled versions of the BERT model, are commonly used as strong baselines for BERT compression. We describe TinyBERT and DistilBERT models in more detail in the next section. Cho and Hariharan (2019) show that distilling from a better (larger and more accurate) teacher does not always lead to a better student model. We see this result as a motivation for using multiple teachers instead of trying to pick the best one to get a better score.

There are several prior studies considering distillation from multiple teachers for Computer Vision (CV) or Natural Language Processing (NLP) tasks. It can be applied to a multi-task or multi-domain setting. For example, Zhang and Peng (2018) combine the knowledge of teachers trained on different tasks, Wu et al. (2019) train teachers on different features extracted from video frames, Ruder et al. (2017) use domain-specific teachers for domain adaptation, and Tan et al. (2019) obtain multilingual machine translation model using teachers pre-trained for each language pair.

Some authors apply multiple teachers without significantly modifying the distillation pipeline, which is closer to our work. Fukuda et al. (2017) propose two ways to utilize multiple teachers in the distillation process: to augment the training data with soft labels provided by different models or to switch the teacher models dynamically at the mini-batch level. Ze et al. (2020) show that averaging the prediction of three teachers trained with different learning rates can improve the score on Question Answering (QA) and Natural Language Inference tasks. Yang et al. (2020) adopted two-stage distillation procedure and showed improvement in several QA tasks. Liu et al. (2019) show the improvement on several tasks from the GLUE benchmark. Additionally, Sau and Balasubramanian (2016) propose to add normally distributed random noise to the logits of the teacher model during distillation, claiming that such procedure is a simulation of learning from multiple teachers.

Besides prediction averaging, multiple teachers can also be utilized to transfer knowledge contained in hidden states or structural relations between examples. You et al. (2017) average soft-labels of multiple teachers and propose to transfer relative dissimilarity among intermediate representations using teacher voting to select the best ordering relationships. Liu et al. (2020) combine soft-labels of multiple teachers with learnable weights, distill structural knowledge between data examples, and transfer intermediate layer representations making each teacher responsible for a specific group of layers in the student network. Both papers relate to the Computer Vision field, both use models with different architectures as teachers, and both show that 5 teachers are better than 3 for their methods (in terms of classification accuracy), but not better for the original knowledge distillation.

To the best of our knowledge, there is no study dedicated to the isolated investigation of the effect that multiple teachers distillation has on the model quality and robustness and of how this effect change with the number of teachers. Importantly, we use models with exactly the same architecture but fine-tuned with different random seeds as teachers.

3 Model Distillation

We briefly describe the formulation of Original Knowledge Distillation procedure (Hinton et al., 2014), two approaches to distill BERT-like models, and one approach to distill the ResNet CV model. Then we describe how multiple teachers get involved in the process.

3.1 Knowledge Distillation

The Original Knowledge Distillation (OKD), proposed by Hinton and co-authors in (Hinton et al., 2014), became an integral part of transferring knowledge from large neural networks to smaller ones. The idea is to train a network called “student” using the task-specific outputs of the so-
Attention distillation involves a "teacher" model as targets. This method combines two losses, namely $L_{CE}$ and $L_{KD}$. With $\lambda$ being a hyper-parameter to control the relative influence of the teacher knowledge transfer.

$$L_{OKD} = L_{CE} + \lambda L_{KD}. \quad (1)$$

$L_{KD}$, Knowledge Distillation loss component, is a metric of proximity between logits of the teacher and student models ($z_T$ and $z_S$ respectively). In this paper, we use the Cross-Entropy variation as this loss:

$$L_{KD} = -\text{softmax}(z_T / t) \cdot \log \text{softmax}(z_S / t), \quad (2)$$

where $t$ is the softmax temperature applied at training time. At the student model inference its softmax temperature is set to 1. $L_{CE}$ is a classic Cross-Entropy loss for label prediction. The following models are modifying this process each in its own way.

### 3.2 TinyBERT

The TinyBERT (Jiao et al., 2019) model has the same general architecture as BERT (Devlin et al., 2018), but has fewer layers and smaller hidden and feed-forward sizes. We experiment with the smallest 4-layer model. The number of attention heads on each Transformer (Vaswani et al., 2017) layer is the same as in BERT (12 heads).

The TinyBERT distillation process involves several loss functions. Assume that the student model has $M + 2$ layers, with 0 and $M + 1$ being the indices of the Embedding layer and Prediction Layer respectively. The Transformer (Vaswani et al., 2017) layers are numbered from 1 to $M$. Every Transformer layer of the student model receives knowledge from the teacher network. Illustrations to this process are presented in Fig. 1. The mapping $g(k)$ between the teacher and student layers is established by a uniform function, in our case the $k$-th layer of the student model learns from $g(k) = 3k$-th layer of the teacher network (BERT BASE). The objective is defined as MSE between the Attention score matrices plus MSE between the outputs of the Transformer layer (after its FFN part). In order for the dimensions of the student and the teacher hidden states to match, a learnable linear transformation is applied to the student states. The resulting loss function looks as follows:

$$L_{\text{Transformer}}(k) = \text{MSE}(H^S_k W_H H^T_{g(k)}) +$$

$$\frac{1}{h} \sum_{i=1}^h \text{MSE}(A^S_{k,i} A^T_{g(k),i}), \quad (3)$$

where $H_k$ is the output of the $k$-th Transformer Layer, $W_H$ is the linear transformation matrix, $h$ is the number of attention heads, and $A^S_{k,i}$ is the Attention matrix of layer $k$ and head $i$. Whether the states are associated with the student or the teacher is indicated by the upper indices S and T, respectively. Similarly, MSE is used to distill the embeddings $E^{S,T}$:

$$L_{\text{emb}} = \text{MSE}(E^S W_e, E^T). \quad (4)$$

For the prediction layer, the Knowledge Distillation loss (2) described above with temperature $t = 1$ is used to adjust the weights of the student.

There are two stages of the TinyBERT distillation process. The first stage is called General Distillation (GD), the large unlabeled corpus (English Wikipedia) is used and the general linguistic knowledge contained in model weights is transferred from teacher to student (the prediction layer is untapped). The following objective is minimized during the distillation process:

$$L_{\text{tiny}} = \sum_{k=0}^M L_{\text{layer}}(S_k, T_{g(k)}). \quad (5)$$

For each layer the loss function is defined by

$$L_{\text{layer}}(k) = \begin{cases} L_{\text{emb}}, & k = 0, \\ L_{\text{Transformer}}(k), & 0 < k \leq M. \end{cases} \quad (6)$$

The second stage is called Task-Specific Distillation (TD) and aims to transfer the task-specific knowledge. It is in fact split into two phases. First, Intermediate Layers Distillation (ILD) is performed...
with the same loss $L_{\text{tiny}}$ (5) as in the General Distillation. Then, Prediction Layer Distillation (PLD) is performed with $L_{KD}$.

The authors apply an augmentation procedure to extend the task-specific training dataset. For every example in the training dataset, $N$ new examples are generated by replacing random words in a sentence with candidates provided by BERT (Devlin et al., 2018) as a language model or by nearest neighbors search in GloVe (Pennington et al., 2014) embedding space. The number $N$ is called the augmentation factor. A detailed description of the algorithm can be found in the original paper (Jiao et al., 2019).

### 3.3 DistilBERT

The DistilBERT (Sanh et al., 2019) model also shares the general architecture with BERT, slightly modifying it by removing the token-type embeddings and the pooling layer. Following the original work, we use 6-layer DistilBERT. The student layers are initialized directly from the teacher network (BERT_{BASE}) weights using the uniform strategy for layer mapping.

In the original work, DistilBERT only obtains knowledge from the teacher BERT through General Distillation, although the authors mention experiments with Task-Specific Distillation on SQuAD dataset (Rajpurkar et al., 2016). For General Distillation, a concatenation of English Wikipedia and Toronto Book Corpus (Zhu et al., 2015) is used as training data. The student model uses both supervised training loss (Masked Language Modeling loss, $L_{\text{MLM}}$) and Knowledge Distillation loss, with logits for the teacher and the student being obtained on Masked Language Modeling (Devlin et al., 2018) task. In addition, the cosine embedding loss $L_{\text{cos}}$ is used, where the cosine distance is calculated between the outputs of the FFN on the last Transformer layers of the teacher and student. Thus, the general DistilBERT model is trained using the following loss:

$$L_{\text{distil}} = L_{\text{MLM}} + L_{\text{KD}} + L_{\text{cos}}.$$  

In the original work, the model is then directly fine-tuned on downstream tasks without the help of a teacher network. In the present work, we experiment with the Task-Specific Distillation applied to DistilBERT. We adopt the two-stage procedure similar to TinyBERT. After obtaining the general model, we perform distillation on task-specific datasets. We experimented with different loss function combinations and found out that the best task-specific performance is achieved when three loss functions are used:

$$L_{\text{TS}}^\text{distil} = L_{CE} + L_{KD} + L_{\text{cos}},$$  

where $L_{CE}$ is the standard Cross-Entropy loss (calculated with ground truth data labels) and $L_{\text{cos}}$ is the same cosine embedding loss as used during the General Distillation. Unlike TinyBERT, we do not split the Task-Specific Distillation stage into two phases (since there is no need to transfer knowledge between deep layers of the networks) and do not use data augmentation.

### 3.4 ResNet

We also tested our method in application to a computer vision task. We use the classic ResNet model described in (He et al., 2016). The key idea behind the ResNet architecture is a residual block which consists of convolutions layers with skip connections, that helps to reduce the gradient vanishing problem which makes possible to build a deeper neural network.

The key difference between knowledge distillation in NLP (both TinyBERT and DistilBERT variations) and CV is the stage of distillation. A CV distillation does not contain the General Distillation phase, hence teacher and student models are not pre-trained on a general task. In our experiments teacher and student models were ResNet variants, namely ResNet-110 and ResNet-20 respectively.

### 3.5 Multiple Teachers (Our Method)

A possible way to provide a student model with more knowledge is to make use of multiple teacher models. This can be achieved by combining predictions, outputs, or hidden states of several models. In the present work, we focus on averaging the logits of all teachers before the final softmax layer. That means that multiple teachers are used exclusively during Prediction Layer Distillation to TinyBERT and during Task-Specific Distillation to DistilBERT. All other stages are conducted with one (primary) teacher. We also use outputs from only one primary teacher model for $L_{\text{cos}}$ loss function during Task-Specific Distillation to DistilBERT to ensure more fair comparison with TinyBERT. Thus, the use of $k$ teachers $\{S_1, \ldots, S_k\}$ is introduced by
slightly changing the $L_{KD}$ formula (2):

$$L_{KD}^k = -\text{softmax}(z^T / t) \times \log \text{softmax}(\sum_{i=1}^k z S_i / (k \cdot t)).$$  \hspace{1cm} (9)

In this paper we obtain different teacher networks for each downstream task simply by fine-tuning BERT\textsubscript{BASE} with different random seeds. We leave the study of other ways of selecting teacher networks to combine as future work.

As for ResNet distillation, since there are no other phases, except the phase of target task distillation, we use multiple teacher distillation technique on it. We again simply fine-tune ResNet-110 with different random seeds to build a set of teachers.

4 Datasets

For evaluation we use a subset of tasks from the GLUE benchmark (Wang et al., 2019) for NLP models. We chose the CoLA task, since the performance drop is the biggest on this dataset for both considered models. The MRPC and SST tasks were chosen in addition to CoLA task due to the average performance drop. Also, the size of MRPC is comparable to CoLA, while SST-2 is a much bigger corpus. Another important feature is that MRPC and SST-2 corpora have test labels publicly available, while the CoLA dataset has not, so below all the results are provided on the development set from this dataset. A brief description of the datasets is provided in this section. We summarize information about the datasets in Table 1. The original results of considered NLP models are presented in Table 2. There are additional datasets in GLUE benchmark, namely: MNLI-m, MNLI-mm, QQP, QNLI, RTE, and STS-B. We provide results on these datasets for reference.

For a computer vision model we chose the classic CIFAR-10 dataset. For the chosen implementation the results, on this dataset are 93.68% and 91.73% for teacher and student respectively\(^1\).

### MRPC

The Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005) contains sentence pairs in the online news domain. The task is to classify whether the sentences in the pair are semantically equivalent (i.e. have the same meaning). We use classification accuracy as the evaluation metric. The test labels are publicly available for this dataset.

### CoLA

The Corpus of Linguistic Acceptability (Warstadt et al., 2019) consists of sentences from linguistic literature. Each example is annotated with a binary label of whether it is a grammatically acceptable English sentence. We evaluate the Matthews correlation coefficient (Matthews, 1975) on the development set only, due to the test labels are not publicly available.

### SST-2

The Stanford Sentiment Treebank (Socher et al., 2013) contains sentences from the movie reviews. We evaluate the classification accuracy on binary sentiment annotation (positive/negative), which can be obtained from publicly available fine-grained five-way sentiment labels for both development and test sets.

### CIFAR-10

CIFAR-10 dataset was presented in (Krizhevsky et al., 2009). It consists of the 50000 training images and 10000 test images scraped from the Internet. They are labeled with 10 categories for classification: plane, car, cat, dog, bird, deer, frog, horse, ship, truck. In this paper, we evaluate the classification accuracy metric in this task. The test labels are publicly available for this dataset.

5 Experiments

In this section we describe the experimental setup and then provide the results and their analysis. All the experiments were performed on a single GeForce RTX 2080 Ti.

---

\(^1\)These results are better than reported in the original paper (He et al., 2016) due to mistakes in the original implementation.
Table 2: Results are evaluated on the test set of GLUE official benchmark datasets. All models are learned in a single-task manner. In the parentheses we provide performance drop (or gain with ‘-’) in comparison to the teacher model marked with *.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dataset</th>
<th>MNLI-m</th>
<th>MNLI-mm</th>
<th>QQP</th>
<th>SST-2</th>
<th>QNLI</th>
<th>MRPC</th>
<th>RTE</th>
<th>CoLA</th>
<th>STS-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT_BASE*</td>
<td>83.9</td>
<td>83.4</td>
<td>71.1</td>
<td>93.4</td>
<td>90.9</td>
<td>87.5</td>
<td>67.0</td>
<td>52.8</td>
<td>85.2</td>
<td></td>
</tr>
<tr>
<td>BERT_SMALL</td>
<td>75.4</td>
<td>74.9</td>
<td>66.5</td>
<td>87.6</td>
<td>84.8</td>
<td>83.2</td>
<td>62.6</td>
<td>19.5</td>
<td>77.1</td>
<td></td>
</tr>
<tr>
<td>DistilBERT</td>
<td>78.9 (5.0)</td>
<td>78.0 (5.4)</td>
<td>68.5 (2.6)</td>
<td>91.4 (2.0)</td>
<td>85.2 (5.7)</td>
<td>82.4 (5.3)</td>
<td>54.1 (12.9)</td>
<td>32.8 (20.0)</td>
<td>76.1 (9.1)</td>
<td></td>
</tr>
<tr>
<td>TinyBERT</td>
<td>82.5 (1.4)</td>
<td>81.8 (1.6)</td>
<td>71.3 (-0.2)</td>
<td>92.6 (0.8)</td>
<td>87.7 (3.2)</td>
<td>86.4 (1.1)</td>
<td>62.9 (4.1)</td>
<td>43.3 (9.5)</td>
<td>79.9 (5.3)</td>
<td></td>
</tr>
</tbody>
</table>

5.1 Distillation Setup

For each dataset we fine-tuned 6 teacher models with different random seeds. Each teacher NLP model is initialized with BERT_BASE uncased version from Huggingface’s Transformers open-source library² (Wolf et al., 2020). We used 30552 as the vocabulary size. Each teacher CV model is initialized as a ResNet-110 model trained on CIFAR-10.

To study the dependence between the number of teachers and the student model scores, we vary the number of teachers from 1 (single teacher distillation) to 6. For each number $k$, we perform experiments with every $k$-combination (unordered) from a 6-teacher set. For instance, for $k = 2$ we have $C_6^2 = 15$ possible combinations. Since distillation procedures for NLP models contain parts where a single teacher is used (ILD for TinyBERT and $L_{cos}$ for DistilBERT), we actually conduct $k$ experiments for each $k$-combination with every teacher from that combination being selected as primary.

For both TinyBERT and DistilBERT, we experiment only with Task-Specific Distillation. As initialization, general models published by the authors are used³. For ResNet models we use an existing

---

²We used Transformers version 2.9.0

³General 4layer-312dim TinyBERT from https://github.com/huawei-noah/Pretrained-Language-Model/tree/master/
implementation\(^4\) to train the models on the CIFAR-10 dataset, since there is no pre-training stage in this model distillation process.

For TinyBERT, we perform Task-Specific Distillation from a single teacher using the original pipeline presented in (Jiao et al., 2019). The only difference is that for the MRPC dataset where a pair of sentences is passed as model input we apply augmentation procedure to both input sentences simultaneously, while (Jiao et al., 2019) leave one of the sentences unchanged. For distillation from multiple teachers, the Intermediate Layers Distillation part remains the same and Prediction Layer Distillation is modified as described above. For each teacher combination, we perform distillation 3 times on MRPC and CoLA with training data files generated by different runs of the augmentation procedure. Since SST-2 has significantly more training data, the results are less dependent on randomness in the augmentation procedure, so we use only one generated file for it.

For DistilBERT, we use our Task-Specific Distillation procedure described above. As we already mentioned, we do not use data augmentation, so we perform distillation 3 times with different random seeds on all datasets to reduce the impact of randomness on the results of our experiments and to have the same number of experiments as with TinyBERT.

5.2 Results
We conducted a series of experiments in order to prove a hypothesis that a distilled model learned from multiple teachers is more robust to a seed choice. We call a model more robust if it has higher worst possible scores, while keeping the best possible scores about the same level.

At first we would like to compare single-teacher models with multiple teacher ones. To do that, we calculate the minimum and the maximum score achieved with each teacher \(k\)-combination. In single teacher mode we simply reuse scores obtained with each teacher included in the combination, while in multiple teacher mode we use all teachers in the combination for the \(L_{\text{KD}}\). Then we average these minimum and maximum scores over all TinyBERT\(\text{distilbert-base-uncased}\) model from Transformers library (https://github.com/huggingface/transformers/tree/master/examples/distillation)\(^4\) https://github.com/akamaster/pytorch_resnet_cifar10

Figure 4: Results for TinyBERT considering time spent on distillation. Shaded: ± one standard deviation.

Figure 5: Results for DistilBERT considering time spent on distillation. Shaded: ± one standard deviation.
combinations for each $k$, obtaining the aggregated measure of models performance.

The scoring results for TinyBERT are provided at Fig. 2, the scoring results for DistilBERT are provided at Fig. 3, while ResNet results are presented at Fig. 6. As one could see the initial hypothesis could be considered true for all the datasets and more than that, the best achievable results are more probable with multiple teacher models for the most models and datasets, with exception of SST-2 for DistilBERT and CoLA for TinyBERT.

One could point out concern regarding the multiple teacher models: these models require significantly larger computational resources to be trained. In order to reduce the potential carbon footprint, we collected additional data for training duration. Since all the training procedures were performed on the same hardware, these measurements could be used for the computational budget comparison. The metrics for TinyBERT are presented at Fig. 4, the metrics for DistilBERT are presented at Fig. 5, while the time consumption for ResNet is presented at Fig. 7. It is readable from the figures that with additional restriction on comparable computational time the hypothesis keeps true, the distilled models are better in the worst case and keep about the same results in the best case, which allows us to call them more robust than the single ones.

6 Conclusion

We showed that the existing distillation process could be improved with the usage of multiple teachers which differ only with random seed initialization. For the NLP models we applied our method to the task-specific distillation, thus improving TinyBERT results. For DistilBERT we modified the original procedure, which led to the improvement in most cases. We also applied the proposed method to ResNet model distillation on the CIFAR-10 task, which led to the quality improvement in all evaluated cases. More than that, the models with roughly the same time consumed by the learning (and distillation) process are better in the worst case, keeping the best results about same level. This keeps true for all the evaluated models and datasets.

As future work, we see an application of the developed technique to the wider variety of models, including the computer vision ones. We hope that our approach can improve the robustness of other modern distillation methods. The additional experiments could be done with more specific tasks, like dialog generation and information retrieval. We hope that our work will foster the research on this topic in the future.

References


Jeff Rasley, Samyam Rajbhandari, Olutunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.


Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu,


615
BERT Embeddings for Automatic Readability Assessment

Joseph Marvin Imperial
National University
Manila, Philippines
jrimperial@national-u.edu.ph

Abstract

Automatic readability assessment (ARA) is the task of evaluating the level of ease or difficulty of text documents for a target audience. For researchers, one of the many open problems in the field is to make such models trained for the task show efficacy even for low-resource languages. In this study, we propose an alternative way of utilizing the information-rich embeddings of BERT models with handcrafted linguistic features through a combined method for readability assessment. Results show that the proposed method outperforms classical approaches in readability assessment using English and Filipino datasets—obtaining as high as 12.4% increase in F1 performance. We also show that the general information encoded in BERT embeddings can be used as a substitute feature set for low-resource languages like Filipino with limited semantic and syntactic NLP tools to explicitly extract feature values for the task.

1 Introduction

Automatic readability assessment is the task of evaluating the level of ease or difficulty of text documents such as web articles, story and picture books, test materials, and medical prescriptions. Often readability levels can be expressed in many forms: discrete values with grade and age levels such as in the Common European Framework of Reference for Languages (CEFR)\(^1\), or with continuous values from a given range such as in the famous Lexile Reading Framework\(^2\). In machine learning setting, this task is most often viewed as a classification task where an annotated set of corpora is trained with its corresponding gold-standard labels evaluated by an expert as mostly done in previous works (Vajjala, 2021; Chatzipanagiotidis et al., 2021; Weiβ and Meurers, 2018; Xia et al., 2016; Reynolds, 2016; Hancke et al., 2012; Vajjala and Meurers, 2012). Recent works have tried testing unexplored resources by utilizing large pre-trained language models such as Bidirectional Encoder Representations or BERT (Devlin et al., 2019) which is based on the attention-driven Transformer architecture by Vaswani et al. (2017) by (a) directly processing the data to the network (Martinc et al., 2021; Tseng et al., 2019) or by (b) using the discrete output of the network via transfer learning (Deutsch et al., 2020) as an additional feature. For these methods, however, evidence of efficacy are only seen in high-resources readability datasets in English. Thus, we propose an alternative way of incorporating the knowledge of large language models such as BERT by combining its information-rich sentence embeddings as a separate feature set for traditional machine learning algorithms with handcrafted linguistic features. We argue that this method is not only low-resource friendly but also preserves the semantic and syntactic information encoded by the attention heads of BERT since the embeddings itself will be used. We show that such information can act as a substitute for languages with limited tools for explicitly extracting semantic and syntactic features where results describe non-significance in difference of performances between models using semantic and syntactic features versus models using BERT embeddings.

2 Previous Work

The first generation of readability formulas and indices date as early as 1920-1940s with the works of Thorndike (1921), Dale and Chall (1948), and Flesch (1948) primarily using surface-based variables such as raw frequencies and average values of sentences and words per document. The process for using such indices requires manual computa-

---

\(^1\)https://www.cambridgeenglish.org/exams-and-tests/cefr/
\(^2\)https://lexile.com/
tion and plugging of values to formulas which can be tedious as the length of a document increases. Likewise, experts argue that considering narrow, surface-based features do not entirely capture the linguistic complexity of a given text (Macahilig, 2015; Collins-Thompson and Callan, 2004; Si and Callan, 2001). Thus, incorporation of deeper, linguistic variables such as a language’s semantics, syntax, morphology, and discourse properties are imperative and worth exploring for the task. To answer this call, the use of handcrafted linguistic features remained the most popular type of input for training readability assessment models through the years. Handcrafted linguistic features are often represented as real-valued numbers serving as potential predictors of the difficulty of reading materials. These features span on a wide range of linguistically motivated factors that base on syntax, semantics, morphology, cohesion, and cognition to name a few. These features also serve as the input in the form of vectors for conventional readability assessment setups using traditional classification-based algorithms. To note, not all linguistic features can be applied or extracted for all languages as some have limited NLP tools suitable for use especially for low-resource languages. Notable works in various languages such as Greek (Chatzipanagiotidis et al., 2021), German (Weiß and Meurers, 2019; Weiß and Meurers, 2018; Hancke et al., 2012), Bangla (Sinha et al., 2012), and Filipino (Imperial and Ong, 2021a, 2020) have used this approach in combination with traditional machine learning algorithms such as Logistic Regression and Support Vector Machines. Likewise, another reason why studies have resorted to the classical approach of model building is that deep neural models are not practical for the task without a large amount of training data.

The advent of large and complex pre-trained language models such as BERT and its variations spawned a handful of studies on how these models fare with the readability assessment tasks. The work of Martinc et al. (2021) on the supervised experiment setup explored directly using English benchmark corpus such as Weebit and OneStopEnglish as input for BERT via transfer learning while Deutsch et al. (2020) explored using the final discrete output of BERT as a feature for the same datasets. Results from both studies show effectiveness of BERT for English data as direct input while no significant improvement is seen when the discrete output itself is used as a feature. While these results are remarkable, BERT’s effectiveness remain a gray area for low-resource languages.

3 Task Definition
We define our task at hand as a supervised learning setup. Given a text document \(d\) where a feature vector \(x = [x_1, x_2, \ldots, x_n]\) is extracted, a model \(M\) is trained using said collection of features \(X\) along with the gold label \(Y\) or expert-identified readability level. The label is relative in form (discrete or continuous) based on how readability levels are categorized for each corpus.

<table>
<thead>
<tr>
<th>Data</th>
<th>Doc Count</th>
<th>Sent Count</th>
<th>Vocab</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSE</td>
<td>367</td>
<td>4,890</td>
<td>17,818</td>
</tr>
<tr>
<td>CCE</td>
<td>168</td>
<td>20,945</td>
<td>78,965</td>
</tr>
<tr>
<td>Adarna House</td>
<td>265</td>
<td>10,018</td>
<td>16,058</td>
</tr>
</tbody>
</table>

Table 1: Data distribution for English and Filipino corpus.

4 Corpus
We describe each corpus used in the study below as well as the statistics and breakdown in Table 1

OneStopEnglish. The OSE corpus is a collection of 567 texts in three different reading levels (beginner, intermediate, and advanced) for adult ESL learners from the MacMillan Education website\(^3\). This corpus was first used in the work of Vajjala and Lučić (2018) and has become one of the most-used benchmark datasets for readability assessment and text simplification in English.

Common Core Exemplars. The CCE dataset contains 168 prose texts from the Appendix B of the Common Core State Standards Initiative (CCSS)\(^4\) for English Language studies and first used by Flor et al. (2013) for readability assessment. The initiative was a project of the National Governors Association and the Council of Chief State School Officers in USA\(^5\). The dataset is divided into three age-range categories: 2-5, 6-7, and 9-12.

Adarna House. The Adarna House corpus is a collection of 265 story books for grades 1-3 from Adarna House Inc.,\(^6\) the largest children’s litera-
ture publisher in the Philippines. This corpus has been used by Imperial et al. (2019); Imperial and Ong (2020, 2021a) for readability assessment in Filipino.

5 BERT Embeddings + Handcrafted Linguistic Features

BERT’s efficacy on a wide range of NLP tasks stems from its implicit capability to encode linguistic knowledge such as hierarchical parse trees (Hewitt and Manning, 2019), parts of speech and syntactic chunks (Liu et al., 2019; Tenney et al., 2019), semantic roles (Ettinger, 2019) as well as entity types and relations (Tenney et al., 2019) to name a few. In view with this, we find such amount of knowledge an extremely valuable resource which can potentially improve performances of readability assessment models especially for low-resource languages if used correctly. Thus, to maximize the potential of BERT for low-resource readability assessment, we propose a combined training of its raw embeddings with handcrafted linguistic feature sets through a concatenation process and feeding them to traditional machine learning algorithms. The embeddings of BERT generated by the multi-head attention layers are information-rich, specifically on semantic and syntactic knowledge (Rogers et al., 2020), due to the nature of its training. We describe our proposed architecture in Figure 1 with a sample Filipino sentence for context.

6 Experiment Setup

For the OSE and CCE corpus in English, we extracted over 155 linguistic features covering lexical diversity and density features, syntactic features based on parse trees, morphosyntactic properties of lemmas, and word-level psycholinguistic features. For the Adarna House corpus in Filipino, we extracted over 54 linguistic features covering traditional surface-based features, lexical features based on POS tags, language model features, morphology based on verb inflection, and orthographic features based on syllable pattern. For the complete and detailed list of features for English and Filipino, please refer to the resources at Vajjala and Lučić (2018) and Imperial and Ong (2020) respectively. The size of the BERT embeddings for all datasets remain equal with a fixed dimension of $H = 768$ since the base version of BERT for English (Devlin et al., 2019) and Filipino (Cruz et al., 2020c; Cruz and Cheng, 2020, 2019) were used. The embeddings and extracted linguistic feature sets were concatenated, for a total of 923 dimensions for combined features for both English datasets and 823 for the Filipino dataset. Recipes for feature extraction were obtained from the studies of Vajjala and Meurers (2016, 2014) for English and Imperial and Ong (2020, 2021a,b) for Filipino. We used the sentence-transformers library by Reimers and Gurevych (2019) with mean pooling option to extract BERT embedding representations for the readability corpora.

For the traditional machine learning algorithms, we used three of the commonly utilized in previous works: Logistic Regression, Support Vector Machines, and Random Forest. Models for each dataset were trained on a 5-fold cross validation procedure. We used weighted F1 as the overall metric for performance evaluation.

7 Results

7.1 Ablation

We compared performances of models on three different setups, (a) linguistic features only, (b) BERT sentence embeddings only, and (c) combined training of the two feature embeddings to gauge the efficacy of the proposed framework.

As described in Table 2, generally speaking, models trained using the proposed combined training of handcrafted linguistic feature sets with contextual BERT embeddings outperform both performances of only using each exclusively on English and Filipino datasets. On average, we note an increase of performance of 2.63% for OSE, 6.23% for CCE, and 12.4% in weighted F1 score for Adarna House across all algorithms. From this, we infer that extracting and incorporating the information-rich embeddings of any readability dataset using BERT to commonly-used linguistic feature sets can substantially improve model performance.

Interestingly, there are a few notable cases reported in Table 2 where BERT embeddings alone outperformed the traditional method of using handcrafted linguistic feature sets as primary input. These cases are evident in the all models utilizing the Adarna House dataset in Filipino with an

---

7Filipino is considered as a low-resource language (Cruz et al., 2020a,b).

---

8We release the script for extracting BERT embeddings at https://github.com/imperialite/BERT-Embeddings-For-ARA
Figure 1: The proposed combined training approach using sentence embeddings from BERT model and extracted handcrafted linguistic feature sets.

<table>
<thead>
<tr>
<th>Method</th>
<th>OSE</th>
<th>CCE</th>
<th>Adarna</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linguistic Features</td>
<td>0.676</td>
<td>0.774</td>
<td>0.389</td>
</tr>
<tr>
<td>BERT Embeddings</td>
<td>0.620</td>
<td>0.747</td>
<td>0.505</td>
</tr>
<tr>
<td><strong>Combined Features (Ling + BERT)</strong></td>
<td><strong>0.732</strong></td>
<td><strong>0.778</strong></td>
<td><strong>0.554</strong></td>
</tr>
</tbody>
</table>

(a) Logistic Regression

<table>
<thead>
<tr>
<th>Method</th>
<th>OSE</th>
<th>CCE</th>
<th>Adarna</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linguistic Features</td>
<td>0.691</td>
<td>0.732</td>
<td>0.414</td>
</tr>
<tr>
<td>BERT Embeddings</td>
<td>0.611</td>
<td>0.826</td>
<td>0.487</td>
</tr>
<tr>
<td><strong>Combined Features (Ling + BERT)</strong></td>
<td><strong>0.704</strong></td>
<td><strong>0.893</strong></td>
<td><strong>0.571</strong></td>
</tr>
</tbody>
</table>

(b) Support Vector Machines

<table>
<thead>
<tr>
<th>Method</th>
<th>OSE</th>
<th>CCE</th>
<th>Adarna</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linguistic Features</td>
<td>0.683</td>
<td>0.842</td>
<td>0.423</td>
</tr>
<tr>
<td>BERT Embeddings</td>
<td>0.439</td>
<td>0.770</td>
<td><strong>0.504</strong></td>
</tr>
<tr>
<td><strong>Combined Features (Ling + BERT)</strong></td>
<td><strong>0.690</strong></td>
<td><strong>0.861</strong></td>
<td>0.467</td>
</tr>
</tbody>
</table>

(c) Random Forest

Table 2: F1 performance via training with (a) Logistic Regression, (b) Support Vector Machines, and (c) Random Forest using handcrafted linguistic features, BERT sentence embeddings, and combined training of both.
average increase of 9.5% weighted F1 scores. From this, we infer that the general semantic and syntactic knowledge implicitly encoded in BERT embeddings as detailed in probing tasks from previous works (Rogers et al., 2020; Hewitt and Manning, 2019; Liu et al., 2019; Tenney et al., 2019) may be significantly more informative than the traditional handcrafted linguistic features for discriminating reading difficulty. Consequently, this poses as a probable and alternative solution for low-resource languages with little to no NLP tools such as a good part-of-speech tagger, stemmer, syntactic parse tree extractor, and morphological analyzer to name a few for manually extracting linguistic information from documents. Since BERT models are trained in an self-supervised manner, the overhead of developing these tools from scratch can be disregarded, at least for readability assessment. We discuss further experiments on this inference in the next section.

7.2 Substituting Semantic and Syntactic Features for BERT Embeddings

To empirically test if BERT embeddings can act as substitute for semantic and syntactic linguistic features for readability assessment, we removed features from the three datasets that assume semantic and syntactic knowledge. For OSE and CCE, we removed 56 features covering part-of-speech densities, lexical richness, type-token densities, and general parse-tree based features. For Adarna, we removed 22 features covering part-of-speech densities, type-token densities, and verb inflection densities. There are no parse-tree based features for Adarna House as there are currently no NLP tools for extracting such feature set for Filipino. The rest of the linguistic features from the datasets denoting other aspects of reading difficulty measurement such as frequency-based features and syllable patterns remain unchanged. Models were retrained using the three selected machine learning algorithms for comparison.

Results of substitution experiments can be found in Table 3. Generally speaking, it is evident that models trained using the combined method still outperforms models using the reduced feature set on the account of CCE and Adarna data. However, we note the 1.2% increase in F1 score on the OSE data. Stemming from this observation, we also note small differences in performances of using the combined features against decreased features. In the CCE corpus, the highest performing model using decreased features obtained 86.9% F1 score which is less 2.4% than using the model with combined features. For the Adarna data, the difference is 6.4%.

To identify if such difference is significant, we used a two-tailed test of difference via Mann-Whitney U using the performance scores of models with combined features and models with decreased features for all datasets. We arrived at a p-value of 0.522 (p >0.5), meaning that the difference of the scores between two groups is not significant. Thus, we conclude that BERT embeddings can be fully used as a substitute for semantic and syntactic features if such information cannot be explicitly extracted from readability data due to the lack of NLP tools and low-resourceness of other languages. To add, since BERT models are trained in an self-supervised manner and there are over 3,000 pretrained models from online repositories, these resources and the proposed combined training method as a viable option.

7.3 Feature Decomposition for Performance Boost

In extending the effort to improve the performance of BERT-enriched readability assessment models and reduce feature size or dimensionality, we resort to the use of feature decomposition to the large feature vector sizes (BERT + linguistic features) via Principal Components Analysis (PCA). PCA works by projecting the overall feature set (often large) to a lower dimensional property while preserving quality and information of features (Hotelling, 1933). We experimented on differing values of variance percentages: 25, 50, 75, 95, and 100 (full, no feature removed). Results of feature decomposition via PCA for each machine learning model are described in Figure 2.

For SVM and Random Forest, all datasets have the highest performances if all features are retained (100 variance percentage). While for Logistic Regression, 75 variance percentage obtained the highest performance with 82.7 F1 score for OSE, 95 variance percentage obtained the highest performance with 83.8% F1 score for CCE, and 100% or full features for Adarna. Thus, we infer that there is no need to perform feature decomposition to find the principal components as the highest-performing

---

9 The distribution of the two groups are of equal variances with p-value of 0.619.

10 https://huggingface.co/models
Table 3: Performances of models via F1 score after retraining with semantic and syntactic handcrafted linguistic features removed to test if information-rich BERT embeddings can act as substitution for such features. Best performing model utilizing combined features from Table 2 appended for comparison.

<table>
<thead>
<tr>
<th>Model w/ Removed Features</th>
<th>OSE</th>
<th>CCE</th>
<th>Adarna</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic Regression</td>
<td><strong>0.744</strong></td>
<td>0.865</td>
<td>0.492</td>
</tr>
<tr>
<td>Support Vector Machines</td>
<td>0.615</td>
<td>0.869</td>
<td>0.507</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.669</td>
<td>0.791</td>
<td>0.431</td>
</tr>
<tr>
<td>Full Model (Ling + BERT)</td>
<td>0.732</td>
<td><strong>0.893</strong></td>
<td><strong>0.571</strong></td>
</tr>
</tbody>
</table>

Figure 2: Decomposing large feature sets on a 25%, 50%, 75%, 95%, and 100% (full) variance percentages using PCA for Logistic Regression, Support Vector Machines, and Random Forest (left to right).

models for OSE, CCE, and Adarna use 100% of the combined feature set (BERT + linguistic features).

8 Conclusion

In this study, we proposed an alternative way of combining information-rich BERT embeddings with handcrafted linguistic features for the readability assessment task. Results from our experiments showed that the method outperforms classical, vanilla approaches in readability assessment using English (OSE and CCE) and Filipino (Adarna) datasets in various machine learning algorithms such as Logistic Regression, Support Vector Machines, and Random Forest. We also demonstrated that the knowledge implicitly encoded in BERT embeddings (semantic and syntactic information) can be used as a full substitute feature set for low-resource languages like Filipino with limited NLP tools to explicitly extract feature values for the task. We are looking forward to the application of our proposed method to other languages struggling with the extraction deep linguistic features to trained readability assessment models. Future directions of the study include deeper exploration of BERT such as isolating extracted embeddings for each of the twelve attention layers.

9 Acknowledgments

We would like to thank Dr. Ani Almario from Adarna House, Dr. Sowmya Vajjala from the National Research Council of Canada, and Dr. Michael Flor from ETS for providing the Adarna, OSE, and CCE datasets respectively.

10 Ethical Considerations

We report that there are no major ethical concerns in the study as it involves no human subjects nor discriminate any identifiable group of people. As for the dataset, for Adarna House, permission was obtained from the publishing house while for the OSE and CCE datasets, it remains open-sourced. As for energy consumption, the study only uses pre-trained BERT models and the authors did not actually perform the pre-training phase itself.

References


Hilaria Cruz, Antonios Anastasopoulos, and Gregory Stump. 2020a. A resource for studying chatino ver-


Heidi B. Macalilig. 2015. A content-based readability formula for Filipino texts. The Normal Lights, 8(1).


Edward L Thorndike. 1921. The teacher’s word book.


Semantic-Based Opinion Summarization

Marcio Lima Inácio and Thiago Alexandre Salgueiro Pardo
Interinstitutional Center for Computational Linguistics (NILC)
Institute of Mathematics and Computer Science (ICMC)
University of São Paulo (USP)
Avenida Trabalhador São-Carlense, 400, 13566-590 São Carlos, SP, Brazil
marcio.lima.inacio@gmail.com, taspardo@icmc.usp.br

Abstract
The amount of information available online can be overwhelming for users to digest, specially when dealing with other users’ comments when making a decision about buying a product or service. In this context, opinion summarization systems are of great value, extracting important information from the texts and presenting them to the user in a more understandable manner. It is also known that the usage of semantic representations can benefit the quality of the generated summaries. This paper aims at developing opinion summarization methods based on Abstract Meaning Representation of texts in the Brazilian Portuguese language. Four different methods have been investigated, alongside some literature approaches. The results show that a Machine Learning-based method produced summaries of higher quality, outperforming other literature techniques on manually constructed semantic graphs. We also show that using parsed graphs over manually annotated ones harmed the output. Finally, an analysis of how important different types of information are for the summarization process suggests that using Sentiment Analysis features did not improve summary quality.

1 Introduction
With the advance of web technologies during the last decades, a numerous amount of textual data is produced constantly, especially User Generated Content within social media and e-commerce domains, which include opinions towards many entities, such as products, organizations, and others. Such quantity of information is virtually impossible for other users to assimilate by their own, which creates a need for automated content selection and summarization methods, originating the Opinion Summarization research area, which integrates both the Text Summarization and Sentiment Analysis areas. (Kim et al., 2011; Liu, 2012).

There are two main types of summaries: extracts, which are composed by copying excerpts from the original texts literally, and abstracts, consisting of new text constructions derived out of the information obtained from the input data. Some authors in the literature call attention to the need of incorporating semantic knowledge into the summarization process (Mani, 2001; Li, 2015; Huang et al., 2020), particularly for producing abstracts, since humans usually create summaries by rewriting, paraphrasing and, mainly, interpreting texts (See et al., 2017; Dohare et al., 2018). This kind of information can be explicitly formalized as semantic representations, such as Predicate Argument Structure (PAS) (Khan et al., 2016, 2018), Abstract Meaning Representation (AMR) (Liu et al., 2015; Liao et al., 2018; Dohare et al., 2018) and others.

There are also arguments promoting the usage of explicit semantic representations to deal with texts from the opinions genre – in which the sentiment expressed by the author is an important information – as this type of linguistic knowledge is able to associate correlated information scattered throughout the texts and also to deal with many different semantic phenomena (such as negations and multi-word expressions) that can alter the interpretation that would be obtained in a word-by-word basis (Cambria, 2013). However, there are, to the best of our knowledge, no works that focus on integrating these representations for opinion summarization.

In this context, this paper focuses on exploring techniques of Text Summarization based on explicit semantic representations upon opinions. As a base to instantiate the semantic knowledge, we used the Abstract Meaning Representation (AMR) (Banarescu et al., 2013), which has already been used in previous research in Text Summarization (Liu et al., 2015; Liao et al., 2018; Dohare et al., 2018) and has also been adapted for the Portuguese language in other former works (Anchiêta and Pardo,
As a result, we present one of the first applications of methods based on explicit semantic representations for Opinion Summarization, showing that the Random Forest machine learning algorithm produced the best summary graphs, according to the metrics applied, outperforming other AMR-based approaches from the literature for manually annotated graphs. The results also indicate, in the same manner as previous works (Liu et al., 2015), that the quality of the AMR graphs used have an impact on the output. The experiments also suggest that using Sentiment Analysis features do not improve summary quality.

In short, the main contributions of this paper are the following:

- Development of four new summarization methods based on the AMR representation with results compatible to the current state of the art on using Semantic Representations for Text Summarization;
- Investigation of how other methods of the literature, not developed initially with opinions in mind, behave in a sentiment analysis context;
- Implementation of a semantic-based opinion summarization tool, which is publicly available\(^1\);
- Analysis of the importance of proposed features for text summarization.

The remaining of the text is organized as follows: in section 2, we present a general notion of the AMR. Later, in section 3, previous works in Text Summarization are outlined with main focus on the ones based on the chosen semantic representation (AMR). Our methods are shown in section 4 followed in section 5 by some important notes about the experiments executed, whose results are then introduced in section 6. Finally, some conclusions and further discussing are made in section 7.

2 Abstract Meaning Representation

Sentences in AMR are represented as rooted directed graphs in which nodes correspond to concepts and edges to relations between them. An example can be seen in Figure 1. Concepts can be of different kinds, e.g framesets (possible-01, recommend-01, ler-01) originated from separate repositories containing predicates and their corresponding arguments. For the English language, this repository is called Propbank\(^2\) (Kingsbury and Palmer, 2002), whilst for Brazilian Portuguese, there is VerboBrasil\(^3\) (Duran and Aluísio, 2015).

The concepts can also be words from the base language, extracted directly, as they are, from the sentence (e, eu, coisa) or derived by some annotation procedure provided in the guidelines, such as the transformation of the comparative “melhor” (better) into a degree relation between “mais” (more) and “bom” (good) (bom, mais). There are also concepts that are specific of the representation formalism (e.g., have-degree-91).

All edges are of two types: argument relations defined for each predicate (ARG0, ARG1, ARG2, ARG3) and the ones determined by the AMR project (op1, op2).

As the original representation was developed specifically for the English language, there is need for adaptations of the guidelines\(^4\) towards other languages. For Brazilian Portuguese, this initiative was made by Anchiêta and Pardo (2018b), followed by Cabezudo and Pardo (2019), dealing with some phenomena that are not so common in English,\(^5\)

\[^1\]The code for each method, as well as for the experiments performed for this paper, is available at: [https://github.com/Superar/SemOpinion3](https://github.com/Superar/SemOpinion3).

\[^2\]Available at: [http://verbs.colorado.edu/verb-index/index.php](http://verbs.colorado.edu/verb-index/index.php).

\[^3\]Available at: [http://143.107.183.175:21380/verbobrasil/](http://143.107.183.175:21380/verbobrasil/).

\[^4\]The original guidelines are available at: [https://github.com/amrisi/amr-guidelines](https://github.com/amrisi/amr-guidelines).
such as indeterminate and hidden subjects.

3 Related Work

There are two main areas related to our work: Summarization based on Semantic Representations and Opinion Summarization, each with a long history in the literature. Thus, we present the works most related to ours, those using the Abstract Meaning Representation and other Opinion Summarization methods.

3.1 Summarization Based On Abstract Meaning Representation

The usage of semantic representations for Text Summarization can be traced back until the 1980s, with the TOPIC system (Reimer and Hahn, 1988). Since then, numerous techniques upon various representations have been developed, such as UNL (Martins and Rino, 2001; Mangairkarasi and Gunasundari, 2012), LNS (Rusu et al., 2009), RSG (Moawad and Aref, 2012) and many others. With a special focus on the AMR representation, used in this paper, there are some works – exclusively for the English language – that are going to be explained more thoroughly below.

The first incorporation of AMR into the Summarization process was conducted by Liu et al. (2015). Their method uses several numerical features to represent nodes and edges, used to calculate a score for the given graph; each attribute has a weight trained through AdaGrad (Duchi et al., 2011), so that graphs similar to gold standard handmade summary graphs are assigned to higher scores. Afterwards, Integer Linear Optimization (ILP) is used upon the original text AMR graph in order to select the subgraph which maximizes the scoring. The authors report a maximum F-score for nodes prediction of 58.7% and of 39% regarding edges.

Later, this technique was enhanced by Liao et al. (2018), as they included new attributes, and a previous sentence clustering step using Spectral Clustering (von Luxburg, 2007), which leverages similarity metrics between each clause, in order to identify the multiple topics in the input. Then, a small amount of sentences are selected from each cluster to proceed with the ILP summarization. They obtained a prediction F-Score of 30.1% for nodes and 9.8% for edges; it is important to note that they used exclusively parsed graphs, which have already been indicated as a factor that harms the results (Liu et al., 2015).

In parallel to these techniques, an unsupervised approach for Summarization using AMR has been developed by Dohare et al. (2018). This method uses the TF-IDF score of each concept in order to select a subset of relevant nodes to serve as a basis for the summary graph. Then, a few rules are used to determine the most important path between each pair of nodes chosen and to, finally, expand these paths by adding edges and nodes according to OpenIE (Banko et al., 2008) triples. This unsupervised method obtained a node prediction F-score of 60.4%, however there is no report for edges.

3.2 Opinion Summarization Approaches

This paper is also included within the Opinion Summarization area, which, to the extent of our knowledge, has no initiative of using explicit semantic representations during this process. Therefore, we focus here on two main works related to ours.

The Opinosis method (Ganesan et al., 2010) uses a graph representation obtained from the text to perform the summarization of opinions. It is important to note that this graph does not convey the semantics of the input, but rather the order with which the tokens occur. A single graph encloses multiple sentences, so that identical tokens are merged into a single node, capturing redundancy. Then, a set of rules are applied upon this graph to find those relevant valid paths, prioritizing nodes which occur in more input sentences. Finally the summary text is composed from these selected paths of tokens with some other additional rules for combining them through conjunctions.

More recently, the Opizer system (Condori and Pardo, 2017) was developed as one of the few projects of Opinion Summarization with a specific focus on the Portuguese language. It comprises two main methods: an extractive (Opizer-E) and an abstractive one (Opizer-A). In Opizer-E, the sentences are first clustered together with respect to their general sentiment polarity and also to which aspect of the product they describe (e.g. the camera, battery, durability, etc.). Then, they are ranked according to an importance score which leverages the sentence position within the document and how the aspects and their corresponding sentiment words are related. The most important sentences are then selected to compose the summary.

In its turn, Opizer-A focuses on the selection of salient n-grams rather than full sentences, clustering them according to the polarity, aspect and con-
tent. Afterwards, a set of representative n-grams is chosen from each cluster with respect to their TF-IDF scores. Lastly, they are used to fill in a predefined textual pattern, resulting in the final abstract.

From this overview of the literature, some methods can be derived and developed, as we present in the following.

4 Methods

As discussed before, the usage of explicit semantic information represents an interesting path for both Text Summarization and Sentiment Analysis, consequently for Opinion Summarization. Thus, we chose the Abstract Meaning Representation as an instance of semantic knowledge. The starting point of our research is, therefore, all the previous works which exploited this representation in the process of summarizing texts (Liu et al., 2015; Liao et al., 2018; Dohare et al., 2018), even if for other genres rather than opinions.

First, some preprocessing is necessary. As AMR is a sentence-level representation, we need to combine multiple sentences into a multi-document graph. This has been done, in this work, similarly to Liu et al. (2015) by merging every node that contains the same concept. Some semantic units – such as named entities or dates – are represented in multiple nodes, which are collapsed before the merging, only allowing the combination of entities with the same overall information (same name or date). This was applied upon all comments about each product, resulting in a multi-document semantic representation of all opinions about some given book or electronic device.

4.1 Rule-based Method

As discussed by Dohare et al. (2018), all steps of their method can be directly applied to multi-document summarization. In this paper, we argue that the usage of the traditional TF-IDF scoring may not be suitable for a multi-document scenario, since it penalizes concepts that occur in a larger range of texts, potentially penalizing important information that is common to a large number of comments. For example, if a lot of comments state that the camera quality of a given smartphone is poor, this information should not be penalized, but rather rewarded. Therefore, we propose a simple variation of this method, by substituting the ranking score form TF-IDF to TF only, i.e. the number of occurrences of each concept across the texts.

4.2 Score Optimization Method

Following the objective of improving the original Dohare et al. (2018) method, we decided to explore the usage of multiple different information in order to enhance the ranking of important nodes.

Similarly to Liu et al. (2015), various features may be applied to represent each node or edge in the graph and then weights can be used to combine these attributes into a salience score for the given element. This idea has been transposed into our work, however we do not use the same features as Liu et al. (2015), as they turn each one into multiple boolean values without any specification of how the thresholds were defined. Thus, we decided to develop our own feature set based on the work of Leskovec et al. (2005), which used graph-based and linguistic features to perform the classification of relevant graph nodes for summarization. Our feature set is presented in Table 1.

As can be seen, our features are almost exclusively focused on the graph structure, excluding four last ones (TF, TF-IDF, Sentiment polarity and Aspect). We decided to keep both TF and TF-IDF, as they can convey some differences, as discussed before, and a combination of both may be interesting. Both Sentiment polarity and Aspect features are common within the Opinion Summarization area (Condori and Pardo, 2017). It is important to report that Leskovec et al. (2005) also states that they used another 118 linguistic features that did enhance the summaries quality, however there is no indication of which these features were, so we decided to not include any other features, as it would be necessary to do a more focused feature engineering research in order to determine which other linguistic information is important and how they should be represented.

After the definition of the feature set, they should be combined into a score for each node. This is accomplished through a linear combination of all attributes with an optimized weight vector, so that important nodes have higher scores. The weights are optimized through the Simulated Annealing method, as it is designed to be capable of escaping from local minima or maxima. To this extent, we used the algorithm developed by Ludermir et al. (2006) for weight optimization in Neural Networks. To obtain the target class of each node (if it should be considered relevant or not), the extractive sum-
<table>
<thead>
<tr>
<th>Feature</th>
<th>Brief description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incoming degree</td>
<td>Number of edges pointing towards the given node.</td>
</tr>
<tr>
<td>Average neighbor degree</td>
<td>Average of the incoming degree values of all neighbors of the given node.</td>
</tr>
<tr>
<td>Degree centrality</td>
<td>Proportion of nodes from the graph that are connected to the given node.</td>
</tr>
<tr>
<td>Eigenvector centrality</td>
<td>Recursive method that assigns higher values to nodes that have a higher incoming degree, taking into account that more important nodes have a stronger “voting power”. (Newman, 2010)</td>
</tr>
<tr>
<td>Pagerank</td>
<td>Similar to the Eigenvector Centrality, but also considering the amount of outgoing edges of an important node. (Brin and Page, 1998)</td>
</tr>
<tr>
<td>HITS</td>
<td>This method results into two coefficients: hub and authority. “Hub” nodes are those which point out to multiple “authority” nodes and vice versa. The scores are calculated jointly. (Kleinberg, 1999)</td>
</tr>
<tr>
<td>Closeness</td>
<td>The importance of a given node is computed as the inverse of the average distance of the given node to all others in the graph. (Newman, 2010, p.181)</td>
</tr>
<tr>
<td>Betweenness</td>
<td>This is calculated as the fraction of shortest paths between each pair of vertices that go through the given node. (Newman, 2010, p.186)</td>
</tr>
<tr>
<td>Local clustering coefficient</td>
<td>Defined as the proportion of triangles formed between the given node and its neighbors among all triangles possible. (Newman, 2010, p.202)</td>
</tr>
<tr>
<td>Depth</td>
<td>Defined as the minimum distance between the root of the graph and the given node.</td>
</tr>
<tr>
<td>TF</td>
<td>Term frequency. Indicates the number of occurrences of a concept over all input comments.</td>
</tr>
<tr>
<td>TF-IDF</td>
<td>The original scoring used by Dohare et al. (2018). It rewards concepts that occur with a high TF value, but penalizes those that are too common throughout a given corpus.</td>
</tr>
<tr>
<td>Sentiment polarity</td>
<td>Three boolean attributes, indicating if the sentiment of the given concept is either positive, negative or neutral according to the OpLexicon (Souza et al., 2011).</td>
</tr>
<tr>
<td>Aspect</td>
<td>Manual aspect annotation provided by Condori et al. (2015) within the OpSums-PT corpus. This boolean feature indicates if a given concept represents an aspect of the main product or not.</td>
</tr>
</tbody>
</table>

4.3 Machine Learning Methods

As there is a new whole set of features developed (Table 1), they can be used as a training input for some Machine Learning algorithms aiming at clas-

sifying nodes as relevant or not, similarly to the original work of Leskovec et al. (2005). The algorithms explored in this work are the following: SVM (originally used by Leskovec et al. (2005)); Decision Trees and Random Forest, which are straightforward interpretable algorithms; and Multi-layer Perceptron, a simple Neural Network method, which is a popular research area currently, specially through Deep Learning<sup>5</sup>. From the selected nodes, the same original rules can be used upon them to

---

<sup>5</sup>We, however, do not explore Deep Learning methods, as they usually require high quantities of data, which we do not have available.
create the final summary graph.

We also explored another machine learning based approach in order to overcome the limitations of the set of rules used, as they prioritize the first sentences of the corpus, which may not be ideal in a multi-document setting. Therefore, one can use the concept of Levi Graphs to also include the relations into the classification problem. Levi Graphs are AMR representations within which the relations (edges) are turned into nodes connected to each original extreme of the edge (Beck et al., 2018). In this way, the relations can also be represented as a feature vector and, consequently, they can be classified directly by the algorithms, dismissing the use of rules for this purpose.

As the classification of edges can lead to disconnected graphs, which would be against the representation guidelines, the largest connected component is selected as the final summary graph.

5 Experiments Setup

As a source of opinions, we used the OpiSums-PT corpus6 (Condori et al., 2015), comprising the opinions from ReLi (Freitas et al., 2014) (concerning 13 books) and also comments about 4 electronic products, obtained from the Buscape e-commerce website7. This corpus also comprehends 10 summaries for each product, 5 extractive and 5 abstractive, created by 14 human specialists. Thus, the corpus contains 171 opinionative documents with a total of 1,502 sentences.

Following the AMR guidelines for the Portuguese language, 404 sentences from the corpus were manually annotated8. Two products of the OpiSums-PT corpus have been fully annotated into AMR to this date. Thus, in this work, they are used as a gold standard test set. In order to compare how the quality of the graphs used with each method interfere in the results, we used an automatic AMR parser (Anchiêta and Pardo, 2018a) to annotate two similar products9, resulting in a total of 4 products in the test set.

We also made use of the parser in order to create an artificial training corpus for the proposed supervised approaches (via score optimization or Machine Learning) from the 13 remaining products of the OpiSums-PT, resulting in 65 examples (pairs of opinions graphs and their corresponding summary graph), with a total of 3,290 AMR sentences, from which 388 are considered relevant to compose the summaries.

Another important resource used is the B2W-Reviews01 corpus10 (Real et al., 2019), which was used, in conjunction with the ReLi (Freitas et al., 2014) corpus (excluding the two products used for the testing) to calculate the Document Frequency term in the TF-IDF counts.

For evaluation, we adopted two metrics from the AMR research community: Smatch (Cai and Knight, 2013) and SEMA (Anchiêta et al., 2019), which are used to compare two AMR graphs (the automatic summary graph and its corresponding gold standard manually annotated graph). As the OpiSums-PT corpus provides 5 summaries for each product, all of them have been used to evaluate the summarization output, however, due to lack of space, we report only the average of these values.

We also would like to call attention to the fact that the Simulated Annealing optimization is sensitive to its random initialization, so it was evaluated by running the same experiment 10 times with different starting weights. The values reported in this paper are the average upon all experiments.

6 Results

The results for every experiment executed are shown in Table 2. For fairness, we compare our methods to other AMR-based summarization approaches, since there is no robust Natural Language Generation tool from AMR to text in Portuguese.

As can be seen, the Random Forest algorithm using the Machine Learning method without Levi graphs, proposed in this work, had the best results among the manual annotated comments, resulting in a Smatch improvement of 15% from the best literature method (Liao et al., 2018). Meanwhile, the Integer Linear Programming based ones (Liu et al., 2015; Liao et al., 2018) were the best for the parsed sentences, indicating that their method is more robust with regards to the quality of the input graphs. The quality of the graphs is, how-

---

6Available at: https://sites.google.com/icmc.usp.br/opinanado.
7Available at: https://www.buscape.com.br.
8Available at: https://github.com/nlcl-nlp/AMR-BP.
9The manual annotated products are the Iphone 5 smartphone and the “O Apanhador no Campo de Centeio” book, similarly we automatically parsed the comments for the Galaxy SIII smartphone and the “O Outro Lado da Meia Noite” book.

10Available at: https://github.com/b2wdigital/b2w-reviews01.
Table 2: Summarization results

<table>
<thead>
<tr>
<th>Method</th>
<th>Smatch</th>
<th>SEMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liu et al. (2015)</td>
<td>0.32</td>
<td>0.22</td>
</tr>
<tr>
<td>Liao et al. (2018)</td>
<td>0.40</td>
<td><strong>0.32</strong></td>
</tr>
<tr>
<td>Dohare et al. (2018)</td>
<td>0.38</td>
<td>0.29</td>
</tr>
<tr>
<td>Rule-based method</td>
<td>0.34</td>
<td>0.24</td>
</tr>
<tr>
<td>Score optimization method</td>
<td>0.38</td>
<td>0.26</td>
</tr>
<tr>
<td>Machine Learning: SVM</td>
<td>0.38</td>
<td>0.27</td>
</tr>
<tr>
<td>Machine Learning: Decision Tree</td>
<td>0.37</td>
<td>0.24</td>
</tr>
<tr>
<td>Machine Learning: Random Forest</td>
<td><strong>0.46</strong></td>
<td>0.27</td>
</tr>
<tr>
<td>Machine Learning: MLP</td>
<td>0.38</td>
<td>0.29</td>
</tr>
<tr>
<td>Machine Learning with Levi graphs: SVM</td>
<td>0.33</td>
<td>0.21</td>
</tr>
<tr>
<td>Machine Learning with Levi graphs: Decision Tree</td>
<td>0.32</td>
<td>0.21</td>
</tr>
<tr>
<td>Machine Learning with Levi graphs: Random Forest</td>
<td>0.32</td>
<td>0.22</td>
</tr>
<tr>
<td>Machine Learning with Levi graphs: MLP</td>
<td>0.33</td>
<td>0.20</td>
</tr>
</tbody>
</table>

We do, however, highlight that this does mean that the attributes set is not fit to the problem, since it worked well with the Random Forest algorithm. Thus, the research of other types of weight optimization techniques, such as Genetic Algorithms as used in the work by Khan et al. (2018), may lead to better results. It is also expected that the inclusion of other linguistic features may improve the outcome (Leskovec et al., 2005).

From the optimized weights, however, we can see how important each of the developed features are. This is accomplished by calculating the correlation of the absolute value for the weights with the evaluation metrics used. This result is shown in Figure 2, in which the more intense the green color is, the more important its corresponding feature is.

As can be seen in the heatmap, the most important features to be noticed are the average neighbor degree, the eigenvector centrality and the depth, with some attention to the clustering coefficient feature. On the other hand, the most detrimental features are those which are assigned to deeper shades of pink. There are not much features which can be considered effectively harmful to the quality of the summaries – i.e. there are no features which have such a high negative correlation to the evaluation metrics – but it can be observed that both the TF-IDF and the Hubs features do have a stronger negative correlation.

The lighter colors are related to the features that
may not have had an impact over the output quality. The sentiment analysis features (sentiment and aspect, commonly used in Opinion Summarization approaches) do not appear to have a role of importance during the summarization process, even though we are dealing with opinions.

The feature analysis also highlights our previous observation that the feature set comprises other types of important information. Although there was not much difference (using Smatch or SEMA) between the score optimization method and the original one from Dohare et al. (2018), the results were obtained taking into account different features (e.g., eigenvector centrality vs. TF-IDF). This indicates that there is indeed room for improvement by combining these forms of knowledge.

7 Conclusion

The incorporation of semantic knowledge via explicit representations into the summarization process is a fruitful field of research. In this context, this paper focused on the development of methods for opinion summarization that took advantage of the semantics encoded within AMR graphs.

The experiments indicate that our Machine Learning approach, using the Random Forest algorithm, produced better summary graphs compared to other AMR-based summarization techniques. We also confirm the findings of Liu et al. (2015), stating that the quality of the input graphs used is important to ensure the creation of good summaries. An analysis of the feature set also suggests that using sentiment and aspect information does not actually improve the results.

As future work, we recommend the further development of the feature set, since it has been shown to be effective for a Machine Learning approach, including mainly new linguistic features, as originally proposed by Leskovec et al. (2005). We also advocate that exploring other optimization techniques can be an important path of investigation, such as the Genetic Algorithm approach used by Khan et al. (2018).

There is also a significant improvement to be done during the merging of sentence graphs into a single document representation. Currently, only the concept labels are taken into account, however the usage of coreference resolution can be a valuable enhancement towards the creation of a consistent multi-sentence representation (Liao et al., 2018; Dohare et al., 2018; O’Gorman et al., 2018).

Acknowledgments

This research was carried out with the support of Coordination for the Improvement of Higher Education Personnel (CAPES #88887.355305/2019-00). We also had the support of the OPINANDO project¹⁹ (PRP #668) and the POeTiSA project²⁰ of the Center for Artificial Intelligence (C4AI)²¹ (IBM and FAPESP #2019/07665-4).

¹⁹https://sites.google.com/icmc.usp.br/opinando/
²⁰https://sites.google.com/icmc.usp.br/poetisa
²¹https://c4ai.inova.usp.br/
References


Using Collaborative Filtering to Model Argument Selection

Sagar Indurkhya
Massachusetts Institute of Technology, Cambridge MA 02139, USA
indurks@mit.edu &
Virtualitics, Inc

Abstract

This study evaluates whether model-based Collaborative Filtering (CF) algorithms, which have been extensively studied and widely used to build recommender systems, can be used to predict which common nouns a predicate can take as its complement. We find that, when trained on verb-noun co-occurrence data drawn from the Corpus of Contemporary American-English (COCA), two popular model-based CF algorithms, Singular Value Decomposition and Non-negative Matrix Factorization, perform well on this task, each achieving an AUROC of at least 0.89 and surpassing several different baselines. We then show that the embedding-vectors for verbs and nouns learned by the two CF models can be quantized (via application of k-means clustering) with minimal loss of performance on the prediction task while only using a small number of verb and noun clusters (relative to the number of distinct verbs and nouns). Finally we evaluate the alignment between the quantized embedding vectors for verbs and the Levin verb classes, finding that the alignment surpassed several randomized baselines. We conclude by discussing how model-based CF algorithms might be applied to learning restrictions on constituent selection between various lexical categories and how these (learned) models could then be used to augment a (rule-based) constituency grammar.

1 Introduction

In learning a language, a child solves many difficult puzzles using limited input data (Berwick et al., 2011; Piattelli-Palmarini and Berwick, 2012; Lasnik and Lidz, 2017). One such puzzle involves the child deciding whether a given verb can take a particular noun as its complement when the child has never previously observed that verb and that noun co-occur in a sentence. To illustrate this puzzle, let us consider an example - suppose the child learner has heard the following sentences:

(a) “I lied and and said that I would not smash the windshield.”
(b) “The robber was not planning to smash every plate.”
(c) “I want to take this hammer and smash the precious vase!”
(d) “They are going to shatter the windshield!”
(e) “The boy who was busy staring at his phone will trip over and shatter his mother’s favorite plate.”
(f) “Did you shatter the blue vase?”
(g) “She knew that Susan would break her expensive new windshield.”
(h) “I saw the man in the red sweater break the delicate plate.”

Now suppose that the child learner has never heard a sentence in which the verb “break” takes the noun “vase” as its complement. How should the child decide whether the following production is licit?

(i) “He is going to trip over and break the vase!”

One strategy that the child might employ is as follows. First the child observes that the verbs “smash” and “shatter” behave similarly by noting that both verbs can select any of the three nouns “windshield”, “plate” and “vase” as a complement (see sentences (a-c) for “smash” and sentences (d-f) for “shatter”). Then the child observes that the verb “break” appears to be similar to the verbs “smash” and “shatter” by noting that the three verbs can select “windshield” and “plate” as complements (see sentences (a, d, g) for “windshield” and sentences (b, e, h) for “plate”). On the basis of these observations, the child may decide that if the verb “break” really is similar (semantically) to “smash” and “shatter”, then “break” should also be able to select “vase” as a complement, just as “smash” and “shatter” can. Likewise, the child observes...
that the nouns “windshield” and “plate” behave similarly, as they can both be taken as complements by the three verbs “smash”, “shatter” and “break”, and that the noun “vase” appears similar to “windshield” and “plate” in so far as the three nouns can be taken as the complement by the two verbs “smash” and “shatter”. This is a second line of observations that the child may use to support their conclusion that sentence (i) is a licit production. The strategy outlined above is a simplified illustration of how a Collaborative Filtering (CF) algorithm (reviewed in §2), which uses evidence of how related verbs and related nouns behave, can be used to infer whether a given verb can take a given noun as a complement.

The goal of the present study is to evaluate whether CF algorithms, a widely used method in artificial intelligence for developing recommender systems (Cacheda et al., 2011; Jalili et al., 2018), can be used to accurately model argument selection based on co-occurrence data obtained from a large English corpus (see §3 for details). The results of the experiments presented in this study (see §4) suggest that model-based CF algorithms perform well on the task of recommending which nouns a given verb can or cannot select as a complement, achieving AUROC of between 0.89 and 0.90 and surpassing a number of different baselines; notably, we found during model selection that the number of latent factors (and thus the size of the embedding vectors learned by these CF algorithms) was relatively small (at most 6 latent factors) as compared to the number of distinct verbs and nouns appearing in the analyzed co-occurrence data. Furthermore, we found that when we used k-means clustering to quantize the (per-verb and per-noun) embedding vectors learned by these CF algorithms, using the cluster of a particular verb or noun as a proxy for the verb or noun itself (as opposed to a distinct embedding vector per verb or per noun) yielded minimal loss of performance (< 1%) even when we used a relatively small number of verb and noun clusters. (See §5 for details) Finally, our results suggest that model-based CF algorithms should be considered for use in modeling the inferences a language learner makes when considering problems of argument selection (see §6 for discussion).

2 A Review of Collaborative Filtering

Given a (finite) set of users, a (finite) set of items, and information about the ratings assigned by users to items (encoded in a user-item rating matrix), the task of a recommender system is to predict whether a given user would select a given item, or what rating the user would assign to the given item - these predictions can then be used to generate a list of recommended items for the given user (Bobadilla et al., 2013). This study takes the users to be predicates (i.e. lexical verbs) and the items to be the arguments (i.e. common nouns) that a predicate may select as its complement (i.e. object).

Content-based recommendation algorithms exploit similarities between the features associated with each item - e.g. a content-based recommendation algorithm would predict which arguments a given predicate can select by evaluating the semantic and syntactic features associated with the argument (Balabanović and Shoham, 1997; Lops et al., 2011). In the case that we do not have access to features associated with the items, we may employ Collaborative Filtering (CF) recommendation algorithms, which consider both similarities between the users and/or similarities between the items.\footnote{See (Herlocker et al., 2004) and (Su and Khoshgoftaar, 2009) for surveys that review Collaborative Filtering algorithms; see (Burke, 2002, 2007) for a review of hybrid recommendation algorithms that combine aspects of content-based and CF algorithms.} CF algorithms are traditionally divided into two groups, memory-based CF algorithms and model-based CF algorithms, which we will now describe.

Memory-based CF algorithms assume that similar users select similar items and assign similar ratings, with two users considered to be similar if they tend to assign similar ratings to items. Memory-based CF algorithms work by identifying a set of users who are similar to the target user – e.g. via the k-Nearest-Neighbor (KNN) algorithm, using the Pearson correlation coefficient as a similarity measure – and then predicting the rating the target user would assign to a given item by considering the ratings assigned by similar users, respectively weighted by the degree-of-similarity of each user to the target user (Schafer et al., 2007).\footnote{Memory-based CF algorithms can work to identify similar users, referred to as user-based collaborative filtering, or alternatively, work to identify similar items, which is referred to as item-based collaborative filtering (Sarwar et al., 2001).}

Although Memory-based CF algorithms are widely used in practice, they face difficulty with respect to: (a) scaling as the sets of users and items grow, and (b) dealing with a sparse user-item interaction matrix (Adomavicius and Tuzhilin, 2005).
Model-based CF algorithms, in which a predictive model is learned, address the aforementioned difficulties of Memory-based models. In particular, latent factor models such as Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF) employ dimensionality-reduction in which user and item profiles (i.e. the rows and columns of the user-item rating matrix) are embedded in a lower-dimensional space in which latent relationships between users and items become more explicit (Hofmann, 2004; Koren et al., 2009; Lü et al., 2012). In this way, latent factor models address two weaknesses of memory-based CF algorithms, scalability and sparsity, and for this reason the present study employs (latent factor) model-based CF algorithms.

3 Deriving a Verb-Noun Rating Matrix from Corpus Data

This study employs an English corpus, the Corpus of Contemporary American English. The COCA is a 385 million word corpus of (late 20th century and early 21st century) English derived from several domains including spoken language, fiction, magazines, newspapers, and academic articles (Davies, 2009, 2010).

The corpus was preprocessed as follows. First, we tokenized and segmented the corpus text into sentences. We then annotated the tokens in each sentence with Part-of-Speech (POS) tags. Sentences without at least one verb and one common noun were then discarded. Finally we lemmatized the tokens in each sentence using the TextBlob python library. After preprocessing the COCA corpus, the text consisted of 75584272 words (not counting punctuation markers or numbers) segmented into 10087753 sentences, with 34329 distinct verb lemmas (derived from 60963 distinct lexical verbs) and 89541 distinct noun lemmas (derived from 10957 distinct common nouns).

Next, we derived a verb-noun co-occurrence matrix from the processed corpus as follows. The set of distinct (lexical) verb lemmas and the set of distinct (common) noun lemmas were indexed (using lexicographic ordering) as \( \{v_1, v_2, v_3, \ldots, v_k\} \) and \( \{n_1, n_2, \ldots, n_l\} \) respectively. Then the entry at row \( i \) and column \( j \) in the verb-noun co-occurrence matrix has value equal to the number of sentences in the (processed) corpus in which the (lemmatized) verb \( v_i \) and the (lemmatized) noun \( n_j \) co-occur, subject to the following constraints:

1. no other noun or verb appears between \( v_i \) and \( n_j \);
2. the verb \( v_i \) must appear in the WordNet\(^4\) verb database (Fellbaum, 1998);
3. the noun \( n_j \) must appear in the WordNet noun database (Miller, 1990, 1998);
4. for a sentence to be counted: (a) the verb \( v_i \) and the noun \( n_j \) must be the last verb and the last noun (respectively) in the sentence, with the verb preceding the noun; (b) neither a pronoun, a punctuation marker (e.g. comma, quotation mark, parenthesis) nor any of the tokens \{ who, what, that, by \} may appear between the verb and the noun.

To compensate for noise in the data, we required that a (verb, noun) pair must appear at least twice to be considered; thus, any entry in the co-occurrence matrix that has a value less than 2 was set to 0. Given our goal of predicting novel (verb, noun) pairings (i.e. where the verb serving as predicate may select the noun as an argument in complement position), we restricted our study to verbs and nouns for which there was evidence (in the corpus data) of co-occurrence with different nouns and verbs respectively. To this end, we removed rows corresponding to verbs that do not co-occur with at least two distinct nouns, and we removed rows corresponding to nouns that do not co-occur with at least two distinct verbs. After this process of reducing the verb-noun co-occurrence matrix, there remained 4172 rows, each corresponding to a distinct predicate lemma, and 12601 columns, each corresponding to a distinct argument column; the verb-noun rating matrix has a total of 716861 non-zero entries. In this way, we computed the verb-noun co-occurrence matrix from the (processed) COCA corpora.\(^5\)

Finally, we constructed the verb-noun rating matrix, which serves as the input to the model-based CF algorithms.\(^6\) We derived a distribution of verb occurrences and a distribution of noun occurrences, and from these two distributions we computed a joint distribution. (See Figure 1) We then used this joint distribution to compute the expected

---

\(^3\)We used the POS-tags employed in annotating the PennTreebank.

\(^4\)See (Miller et al., 1990; Miller, 1995; Miller and Fellbaum, 2007) for reference on the WordNet database.

\(^5\)The verb-noun co-occurrence matrix is stored as a list of three-tuples of (verb, noun, counts) for all (verb, noun) pairs for which counts is non-zero.

\(^6\)This matrix corresponds to the user-item rating matrix in the terminology of collaborative filtering algorithms.
number of counts for each (verb, noun) pairing. We assigned a rating to each (verb, noun) pairing for which there were a non-zero entry in the co-occurrence matrix. Whether a (predicate, argument) pairing was rated high or low corresponded to whether that predicate co-occurred with that argument more or less often than would be expected by chance. The ratings were thus assigned as follows: (i) a high rating, which has numerical value 2, was assigned if the value in the co-occurrence matrix was greater than the expected number of counts; (ii) a low rating, which has numerical value 1, was assigned otherwise.\footnote{7}

4 Experiment

The experiment detailed in this section addresses the question of whether model-based CF algorithms can be used to accurately predict which arguments (i.e. common nouns) a predicate (i.e. a lexical verb) may select.\footnote{8} We evaluated two different latent factor models, SVD and NMF.\footnote{9}

4.1 Methodology

To train a model-based CF algorithm, we employed nested $k$-fold cross-validation with shuffling, with the outer loop used to evaluate trained models, and the inner loop used for model selection (hyperparameter tuning) and model fitting (i.e. training). The outer loop consists of a 5-fold cross-validation loop, with 20\% of the data (i.e. entries in the verb-noun rating matrix) held out as a test data set, and the remaining data used for training and validation. The inner loop consist of a 5-fold cross-validation loop, with 20\% of the data held out as a validation set, and the remaining data used for training. Model selection for both SVD and NMF consisted of opti-
mizing the hyperparameter for the number of latent factors, \( n_f \in [4, 21] \).\textsuperscript{10} Models were evaluated for selection by computing the mean average error (MAE), a commonly used metric used for evaluating model-based CF algorithms. The output of a trained model thus consists of: (i) a mapping between predicates and embedding vectors of length \( n_f + 1 \); (ii) a mapping between arguments and embedding vectors of length \( n_f + 1 \); (iii) a matrix with \( n_f + 1 \) and \( n_f + 1 \) columns.

4.2 Results

We computed four baselines that, given a (predicate, argument) pairing \((p, a)\) in the test set, make the following predictions:

- The \textit{pred.med} and \textit{pred.avg} baselines predict the median and mean values (respectively) of entries with predicate \( p \) (in the training set).
- The \textit{arg.med} and \textit{arg.avg} baselines predicts the median and mean values (respectively) of entries with argument \( a \) (in the training set).

The two trained collaborative filtering models and the four baselines each produce, for a given predicate and argument, a continuous value that we interpret as being a high or low rating based on whether is above or particular threshold value or not. We thus evaluated the performance of each model by computing the Area Under the Receiver Operating Characteristic curve (AUROC), which is presented in Figure 2.\textsuperscript{11} Notably, the two CF algorithms, SVD and NMF, achieved an AUROC of 0.90 and 0.89 respectively, and both models outperformed each of the baselines - this suggests that both of these models perform well on the task of predicting which arguments a predicate can take as its complement. See Table 1 for examples of model predictions.

5 Analysis

Having seen that the two model-based CF algorithms performed well on the task of predicting which arguments a predicate is likely to select as its complement, we now turn to considering whether these CF models encode some knowledge of lexical-semantics. Lexical verbs may be said to cluster together into classes, with lexical verbs in the same class sharing similar syntactic and semantic properties (see the verb-classification established in (Levin, 1993)). The model-based CF algorithms that were evaluated in §4 produced distinct embedding vectors for each argument and for each predicate - this section analyzes whether these embedding vectors for verbs and nouns can be clustered into disjoint groups (of embedding vectors for verbs and nouns respectively). We will restrict our attention to the embedding vectors learned by the SVD and NMF models with median performance (with respect to AUROC).

For each of the two trained CF models, we used the \textit{k-means} algorithm (Lloyd, 1982; Forgey, 1965; Jain, 2010) to group the embedding vectors for predicates and arguments into disjoint clusters of predicates and arguments respectively. The \textit{k-means} algorithm was parameterized by the number of clusters the input (embedding) vectors should be grouped into. Let \( k_p \) denote the number of clusters the predicate embedding vectors are grouped into, and let \( k_a \) denote the number of clusters the argument embedding vectors are grouped into. A grouping of the predicates and

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Recommended</th>
<th>Not Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin</td>
<td>dynasty, impeachment, simulation</td>
<td>mouth, side, weapon</td>
</tr>
<tr>
<td>challenge</td>
<td>doctrine, misconception, paradigm</td>
<td>mother, eye, place</td>
</tr>
<tr>
<td>destroy</td>
<td>battleship, habitat, rival</td>
<td>lot, week, word</td>
</tr>
<tr>
<td>elect</td>
<td>bishop, spokesperson, successor</td>
<td>life, place, world</td>
</tr>
<tr>
<td>perform</td>
<td>masterpiece, pushup, somersault</td>
<td>air, mother, town</td>
</tr>
<tr>
<td>sell</td>
<td>denim, postage, sushi</td>
<td>circumstance, editor, father</td>
</tr>
<tr>
<td>sit</td>
<td>cafeteria, parliament, veranda</td>
<td>employee, history, justice</td>
</tr>
<tr>
<td>spread</td>
<td>fluid, manure, paperwork</td>
<td>break, friend, point</td>
</tr>
</tbody>
</table>

Table 1: Examples of arguments that the SVD model with median performance (as measured by AUROC) does or does not recommend for selection as a complement for the listed predicate.

\textsuperscript{10}Both models were trained over 350 epochs. For SVD we used a learning rate of 0.005 and a regularization rate of 0.02; for NMF we used a regularization rate of 0.06.

\textsuperscript{11}See (Fawcett, 2006) for discussion on interpreting the Receiver Operating Characteristic curve.
arguments using particular values of $k_p$ and $k_a$ respectively is referred to as a model-grouping. We ran $k$-means clustering to compute model-groupings using $k_p \in \{10, 15, 20, \ldots, 55\}$ and $k_a \in \{10, 15, 20, \ldots, 100\}$, with clustering run five times for each selection of $k_p$ and $k_a$.

We next evaluated how well a model-grouping could be used to predict whether a given predicate (lexical verb) selects a particular argument (common noun). Given a particular model-grouping with $k_p$ predicate clusters and $k_a$ argument clusters, for a verb-cluster, $\alpha$, and a noun-cluster, $\beta$, let the $c_{\alpha, \beta}$ be the total number of co-occurrence counts for (verb, noun) pairs in $(\alpha, b)$ that have a high rating, and let $d_{\alpha, \beta}$ be the total number of co-occurrence counts for (verb, noun) pairs in $(\alpha, b)$; next, define the average rating $a_{\alpha, \beta}$ to be $\frac{c_{\alpha, \beta}}{d_{\alpha, \beta}}$, and let us suppose that all (verb, noun) pairs in $(\alpha, \beta)$ are assigned the rating $a_{\alpha, \beta}$. We can then recompute the AUROC for this model-grouping on the test data to evaluate the discriminatory power of the model-grouping; we refer to this metric as the clustered-model AUROC.

We identified model-groupings that achieved near-optimal clustered-model AUROC while using as few predicate clusters and argument clusters as possible. We did this as follows. Let the optimal clustered-model AUROC be the maximal clustered-model AUROC achieved by any model-grouping, and let a model-grouping be called sub-optimal if its clustered-model AUROC is less than 99% of the optimal clustered-model AUROC. Then a model-grouping is called near-optimal if (i) its clustered-model AUROC is within 1% of the optimal clustered-model AUROC, and (ii) decreasing either $k_p$ or $k_a$ yields a sub-optimal model-grouping. We computed the clustered-model AUROC for each model-grouping and identified model-groupings that were near-optimal. We found that near optimal performance (i.e. $< 1\%$ loss of maximal clustered-model AUROC) could be achieved using a relatively small number of clusters for verbs and nouns (relative to the dimensions of the verb-noun rating matrix). See Table 2 for statistics of both near-optimal model-groupings as well as model-groupings that obtained the optimal clustered-model AUROC.\footnote{See Figure 4 and Figure 5 in the appendix for the distribution of the clustered-model AUROC for each of the model-groupings, for the (median-performing) SVD and NMF models (respectively).}

We also computed a summary statistic, the weighted average entropy of a clustering, that serves as a measure of how well (on average) the various verb and noun clusters in a model-grouping are able to cluster together similarly behaving verbs and nouns. This statistic was computed as follows. Given a particular model-grouping with $k_p$ verb clusters and $k_a$ noun clusters, for a verb-cluster, $\alpha$, and a noun-cluster, $\beta$, the entropy for this pair of verb and noun clusters is:

$$e_{(\alpha, \beta)} = -\frac{c_{\alpha, \beta}}{d_{\alpha, \beta}} \log \left( \frac{c_{\alpha, \beta}}{d_{\alpha, \beta}} \right)$$

The entropy $e_{(\alpha, \beta)}$ measures how much information is needed (on average) to determine whether a randomly selected (verb, noun) pair in $(\alpha, \beta)$ has a high or low rating; note that since there are only two possible ratings (high and low), $0 \leq e_{(\alpha, \beta)} \leq 1$, and we expect a good (verb, noun) cluster-pair to have an entropy value closer to 0 than to 1, as that signifies that the members of the cluster pair are either majority high-rated or low-rated (verb, noun) pairs, which in turn suggests that these (verb, noun) pairs behave in a similar manner and are thus appropriate to cluster together. We then compute the weighted average entropy (also bounded between 0 and 1) of the model-grouping $(k_p, k_a)$ is computed as:

$$W_{(k_a, k_a)} = \frac{\sum_{\alpha, \beta} c_{\beta} e_{(\alpha, \beta)}}{\sum_{\alpha, \beta} d_{\alpha, \beta}}$$

Table 2: Statistics for model groupings that are near-optimal with respect to AUROC on test data. Here $k_p$ is the number of predicate (verb) clusters, $k_a$ is the number of argument (noun) clusters. The two bolded rows at the bottom are the model groupings that had the maximal AUROC on the test data.

<table>
<thead>
<tr>
<th>model</th>
<th>$k_p$</th>
<th>$k_a$</th>
<th>AUROC</th>
<th>$W_{(k_a, k_a)}$</th>
<th>NMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD</td>
<td>55</td>
<td>20</td>
<td>0.847</td>
<td>0.407</td>
<td>0.201</td>
</tr>
<tr>
<td>SVD</td>
<td>40</td>
<td>25</td>
<td>0.847</td>
<td>0.407</td>
<td>0.180</td>
</tr>
<tr>
<td>SVD</td>
<td>35</td>
<td>30</td>
<td>0.846</td>
<td>0.409</td>
<td>0.169</td>
</tr>
<tr>
<td>SVD</td>
<td>30</td>
<td>50</td>
<td>0.846</td>
<td>0.405</td>
<td>0.153</td>
</tr>
<tr>
<td>SVD</td>
<td>25</td>
<td>100</td>
<td>0.847</td>
<td>0.405</td>
<td>0.143</td>
</tr>
<tr>
<td>NMF</td>
<td>55</td>
<td>30</td>
<td>0.860</td>
<td>0.388</td>
<td>0.228</td>
</tr>
<tr>
<td>NMF</td>
<td>40</td>
<td>35</td>
<td>0.860</td>
<td>0.391</td>
<td>0.208</td>
</tr>
<tr>
<td>NMF</td>
<td>35</td>
<td>40</td>
<td>0.860</td>
<td>0.389</td>
<td>0.197</td>
</tr>
<tr>
<td>NMF</td>
<td>30</td>
<td>50</td>
<td>0.859</td>
<td>0.391</td>
<td>0.184</td>
</tr>
<tr>
<td>NMF</td>
<td>25</td>
<td>100</td>
<td>0.862</td>
<td>0.385</td>
<td>0.175</td>
</tr>
<tr>
<td>SVD</td>
<td>55</td>
<td>95</td>
<td>0.854</td>
<td>0.383</td>
<td>0.198</td>
</tr>
<tr>
<td>NMF</td>
<td>55</td>
<td>100</td>
<td>0.868</td>
<td>0.372</td>
<td>0.227</td>
</tr>
</tbody>
</table>
Finally we sought to understand whether the information encoded within the verb embedding-vectors learned by a latent factor model aligns with traditional classification of verbs by their syntactic and semantic properties. To this end, we computed, for both the SVD and NMF models, the Normalized Mutual Information (NMI) between the clusterings of the predicate embedding vectors and the Levin verb classes. The Levin verb classes were restricted to verbs that appear as verbs in WordNet, and (following (Li and Brew, 2008)) we removed verbs that appeared in more than two verb classes. We also compared the NMI between the Levin verb classes against two baselines:

- Baseline A: group the predicates into \(k_p\) clusters of even size, with predicates randomly assigned to the clusters.
- Baseline B: create \(k_p\) clusters and randomly assign each predicate to one of the clusters.

We found the NMF model consistently had a higher NMI score than the SVD model (and thus aligned more closely with the Levin verb classification), and that both the SVD and the NMF model had higher NMI scores than the two baselines (see Figure 3). Note that the Levin verb classes were derived by considering the set of predicate-argument frames that a verb appears in, of which the argument-selection considered in this study (i.e. how a predicate selects an argument that serves as its complement) is a subset; thus, we expected at most a partial alignment between the predicate embedding vectors produced by the model-based CF algorithms and the Levin verb classes.

6 Conclusion

The results of this study suggest that model-based CF algorithms perform well on the task of inferring which (common) nouns a given (lexical) verb serving as a predicate can select as a complement. The two model-based CF algorithms that were evaluated on this task, SVD and NMF, achieved AUROC of 0.90 and 0.89 (respectively), indicating that they have good discriminatory power and surpassing the performance of several baselines. Notably, these two models achieved this level of performance using at most six latent factors, yielding embedding vectors of relatively low dimensionality as compared to the dimensionality of the verb-noun rating matrix from which they were derived. We also found that the embedding vectors yielded by the SVD and NMF models could each be clustered into a small number of disjoint groups with only a minuscule loss of performance (as measured by AUROC). Finally, we observed modest alignment of the verb clusters with the Levin verb classes as compared to several baselines. We believe that the results presented in this study warrant evaluation of whether model-based CF algorithms are suitable for modeling constituent selection, thereby going beyond inferring the arguments a predicate may select as its complement; such a set of CF models for constituent selection could be used to constrain the productivity of the rules that make up a constituent grammar, thereby yielding a system for learning a grammar from noisy corpus data.

Acknowledgments

Three anonymous reviewers are thanked for critically reading the manuscript and providing helpful comments.

References


Marko Balabanović and Yoav Shoham. 1997. Fab:


Recommender Systems.


Appendix

Figure 4: AUROC for each Model-Grouping derived from the median-performing SVD model.

Figure 5: AUROC for each Model-Grouping derived from the median-performing NMF model.
Figure 6: Weighted Average Entropy for each Model-Grouping of the median-performing SVD model.

Figure 7: Weighted Average Entropy for each Model-Grouping of the median-performing NMF model.
Abstract

Recently, domain shift, which affects accuracy due to differences in data between source and target domains, has become a serious issue when using machine learning methods to solve natural language processing tasks. With additional pretraining and fine-tuning using a target domain corpus, pretraining models such as BERT\(^1\) can address this issue. However, the additional pretraining of the BERT model is difficult because it requires significant computing resources.

The efficiently learning an encoder that classifies token replacements accurately (ELECTRA) pretraining model replaces the BERT pretraining method’s masked language modeling with a method called replaced token detection, which improves the computational efficiency and allows the additional pretraining of the model to a practical extent.

Herein, we propose a method for addressing the computational efficiency of pretraining models in domain shift by constructing an ELECTRA pretraining model on a Japanese dataset and additional pretraining this model in a downstream task using a corpus from the target domain.

We constructed a pretraining model for ELECTRA in Japanese and conducted experiments on a document classification task using data from Japanese news articles. Results show that even a model smaller than the pretrained model performs equally well.

1 Introduction

A domain shift problem occurs when using machine learning to solve natural language processing tasks, where the training data (source) domain is differs from the test data (target) domain.

Because downstream tasks fine-tune a model, pretrained models such as BERT (Devlin et al., 2019) can deal with the domain shift problem. To improve accuracy, the additional pretraining of BERT using the target domain corpus and fine-tuning of the additional pretrained models have been used recently. However, the additional pretraining of BERT requires considerable computing resources and therefore cannot be performed easily. Furthermore, a large corpus of the target domain to be used is required, but this corpus is often unavailable in reality.

In this paper, we attempted to address the computational efficiency of pretraining in domain shifting using efficiently learning an encoder that classifies token replacements accurately (ELECTRA) (Clark et al., 2020).

ELECTRA is a pretraining model that uses replaced token detection (RTD) to replace the masked language modeling (MLM) used in BERT. In MLM, the model is trained by estimating the [MASK] and replacing some words in the input sentence with [MASK]. However, only 15% of the words in BERT are replaced by [MASK], which is computationally inefficient: thus, RTD is an improvement on BERT.

RTD provides two models: generative and discriminative, which are based on the idea of generative adversarial network (GAN) (Goodfellow et al., 2014). The discriminator is pretrained by deciding if it replaces each token generated by the generator. RTD is computationally more efficient because it can handle all tokens in training, and the ELECTRA model using RTD performs better than the BERT model of the same size.

In this study, we first built a general small-scale ELECTRA model. Thereafter, we constructed a domain-specific ELECTRA model by additional pretraining it using a small corpus of the target domain. Although the constructed domain-specific ELECTRA model is smaller than BERT-base, we confirmed that it outperforms BERT-base in the
target domain for a document classification task.

2 Method

The corpus used to train pretraining models can be biased (Bai et al., 2020). We can use a pretraining model unique to the domain that the actual system is targeting to increase accuracy in that domain if we construct one. However, most of the current pretraining methods require considerable computational resources to be effective.

In most cases, increasing the amount of computation needed for pretraining, would increase the accuracy of the downstream task; however, when conducting pretraining, considering the accuracy of the downstream task as well as the computational performance is important.

Therefore, herein, we use ELECTRA, that uses RTD, a computationally efficient pretraining method.

ELECTRA models of various sizes evaluated the performance of the downstream tasks considering computational complexity. Experiments on GLUE (Wang et al., 2018), a benchmark for natural language understanding, and SQuAD (Rajpurkar et al., 2016), a benchmark for question answering techniques, were performed. For the same amount of computation, the ELECTRA model outperforms other state-of-the-art natural language processing models. For example, it performs RoBERTa and XLNet with less than 25% of the computational effort.

For further efficiency, ELECTRA-Small, which can be trained in four days on a single GPU, performs better than GPT and requires only 1/30 of the computation.

Building a pretraining model using ELECTRA, which employs RTD, a more computationally efficient pretraining method, and performing additional pretraining on the corpus in the target domain, we can build a model with accuracy comparable to existing pretraining models for document classification tasks with fewer computational resources and less training time; see Figure 1 for an overview.

3 Experiments

3.1 Pretraining Model

We used a program (run_pretraining.py) available on the official GitHub 2, free TPU resources on Google Colaboratory (Colab), and Google Cloud Storage (GCS) to build the ELECTRA model with pretraining in Japanese. We can reduce the training time even more than the GPU using the TPU resources on Colab. Furthermore, TPU on Colab can only input and output data via GCS, so using GCS is necessary.

For the pretraining corpus, we used the full text of Wikipedia in Japanese, which is same as BERT from Tohoku University (Tohoku-BERT). We used Mecab-NEologd for the tokenizer as well as for Tohoku-BERT.

To build the training corpus, preprocess the text, create the vocabulary files, and create the TensorFlow dataset for pretraining, the software on Tohoku-official BERT’s GitHub 3 was used.

3.2 Model Evaluation

The build model was evaluated based on its success on a document classification task in a small domain. We used the Livedoor-news corpus as the evaluation data for fine-tuning. This is a dataset of Japanese news articles from Livedoor-news published by RONDHUIT Inc.

Each document comprises a URL, creation date, a title, and a body text. Here, we labeled the article body numerically according to its category.

We divide the text of an article into training and test data for each of the nine categories, train a model on the training data, perform a nine-value classification task on the test data to predict the article’s category from the text of the article, and assess performance based on the percentage of correct answers.

The numerical labels for each category and number of articles included are shown in Table 1.

3.3 Results

ELECTRA-JP-Small, the model we developed, was pretrained with small size parameters rather than base size parameters, as in Tohoku-BERT. This is because we also consider the computational efficiency of pretraining. In fine-tuning, training is done up to 50 epochs. The trained models are saved for each epoch, and the value with the highest percentage of correct task answers for each model is selected. The results are shown in Table 2.

We perform model comparison experiments using the SentencePiece-based Japanese ELECTRA model (ELECTRA-JP-SentencePiece) released by

---

2https://github.com/google-research/electra
3https://github.com/cl-tohoku/bert-japanese
Table 1: Numerical labels and number of articles in each category.

<table>
<thead>
<tr>
<th>Class</th>
<th>Category</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>dokujo-tsushin</td>
<td>87</td>
<td>696</td>
</tr>
<tr>
<td>1</td>
<td>it-life-hack</td>
<td>87</td>
<td>696</td>
</tr>
<tr>
<td>2</td>
<td>kaden-channel</td>
<td>86</td>
<td>692</td>
</tr>
<tr>
<td>3</td>
<td>livedoor-homme</td>
<td>51</td>
<td>409</td>
</tr>
<tr>
<td>4</td>
<td>movie-enter</td>
<td>87</td>
<td>696</td>
</tr>
<tr>
<td>5</td>
<td>peachy</td>
<td>84</td>
<td>674</td>
</tr>
<tr>
<td>6</td>
<td>smax</td>
<td>87</td>
<td>696</td>
</tr>
<tr>
<td>7</td>
<td>sports-watch</td>
<td>90</td>
<td>720</td>
</tr>
<tr>
<td>8</td>
<td>topic-news</td>
<td>77</td>
<td>616</td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>736</td>
<td>5895</td>
</tr>
</tbody>
</table>

Table 2: Experimental results of fine-tuning. Accuracy is the highest percentage of correct answers for each model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tohoku-BERT</td>
<td>0.8835</td>
</tr>
<tr>
<td>ELECTRA-JP-Small</td>
<td>0.8412</td>
</tr>
<tr>
<td>ELECTRA-JP-SentencePiece</td>
<td>0.8024</td>
</tr>
</tbody>
</table>

Cinnamon AI Inc. as a guide. This model is a pretrained model with the same parameter size as ELECTRA-Small.

The ELECTRA-JP-Small model provides a higher percentage of correct answers than the ELECTRA-JP-SentencePiece model, (Table 2). However, because the model’s parameter size is smaller than the base size, the correct response rate is approximately 4% lower than the Tohoku-BERT model.

3.4 Additional Pretraining

Additional pretraining using this small corpus would enable us to create comparable models; we have been fine-tuning using a domain-specific small corpus and noting the computational efficiency of the ELECTRA models.

Therefore, we experimented to confirm this process. The text of articles from the Livedoor-news corpus, which was used in the previous experiment, was extracted in plain text and used as a single pretraining dataset for the additional pretraining of ELECTRA-JP-Small.

The ELECTRA-JP-Small has already been pretrained for 1 M steps (24 h in training time). We increase the number of steps in this model (ELECTRA-JP-Small-1.25M) by 0.25 M (10 h of training time); see Figure 2 for an overview.

After the additional pretraining, we run the fine-tuning five times on the models, extract the highest correct response rate value from each of the five models, and show the average and highest values in Table 3.

As shown in the table, even with a small parameter size model, we could to confirm that by performing additional pretraining with a small corpus of domain-specific data, we can create an ELECTRA model that outperforms Tohoku-BERT.

3.5 Prediction Time

The ELECTRA model we developed is smaller than the Tohoku-BERT model. This is expected to reduce the training and prediction times during fine-tuning. To confirm this, we measured the training and prediction times for each of the 50 models created during fine-tuning for both ELECTRA-JP-Small and Tohoku-BERT. The results are shown in
3.6 Model Size

Table 4 shows the predictions of the pretraining time up to 1M steps using TPU for different model sizes.

The results in Table 2 are not an exact comparison of model performance because of the performance difference due to the parameter size of the pretraining model. Even with one TPU resource, building an ELECTRA model with the same parameter size as Tohoku-BERT requires approximately a week of training time. As pretraining requires huge computational resources, the larger the model size, the more difficult it becomes to build a pretrained model. The ELECTRA model, which is more parameter efficient for smaller models, the performance was reasonably good even for small size. However, this does not go far enough to override the performance of the different model sizes. To confirm the difference in model performance more accurately, constructing pretraining models of the same size is necessary.

3.7 Effects of Additional Pretraining

From the results in Tables 2 and 3, it is clear that the pretraining model with a small corpus of domain-specific data before the downstream task can im-

---

Figure 2: Overview of additional pretraining.

<table>
<thead>
<tr>
<th>Model</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELECTRA-JP-Small</td>
<td>1d 22hs</td>
</tr>
<tr>
<td>ELECTRA-JP-Base</td>
<td>7d 1h</td>
</tr>
</tbody>
</table>

---

Table 3: Experimental results of fine-tuning after additional pretraining. *Average* is the average of the values obtained by running fine-tuning five times for each of the five models with the highest percentage of correct answers. *Max* is the highest value of the five models’ correct responses.

<table>
<thead>
<tr>
<th>Model</th>
<th>Average</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tohoku-BERT</td>
<td>0.8814</td>
<td>0.8835</td>
</tr>
<tr>
<td>ELECTRA-JP-Small-1.25M</td>
<td>0.8834</td>
<td>0.8864</td>
</tr>
</tbody>
</table>

---

Table 4: Pretraining time for models up to 1M steps. The time in the table is the predicted pretraining time for the model up to 1M Steps.

---

Figure 3.

The average training time for each BERT model was approximately 61.7 s, and for each ELECTRA model was approximately 14.4 s. Moreover, each model in BERT took an average of 160.4 s to predict, and in ELECTRA took on average of 33.6 s. Summing up the time taken by each of the 50 models, the training time for BERT was 51 min 25.1 s and the prediction time was 2 h 13 min 39.7 s, whereas the training time for ELECTRA was 11 min 58.7 s and the prediction time was 27 min 59.7 s. From these results, it can be seen that ELECTRA-JP-Small can train the model in approximately 1/4 the length and predict in approximately 1/5 the length than Tohoku-BERT.
prove the model’s accuracy after fine-tuning.

The larger the parameter size of the model, even for BERT and ELECTRA, the more time it takes to pre-train the model, and thus the more difficult it is to perform additional pretraining. However, for small models, the time required for additional pretraining is much shorter than for base-size models. Therefore, additional pretraining of the ELECTRA-Small model with a small corpus of domain-specific models can achieve better performance with less computation.

4 Related Work

4.1 Masked Language Modeling

MLM masks a certain percentage of words in the input sentence (usually approximately 15%). The masked words are replaced with other words or special tokens, such as [MASK], and the model is assigned to predict the masked words. BERT with such a pretraining method outperforms the conventional language model on many of the downstream tasks. However, because the MLM methods is used to predict masked words, only 15% of the masked words in each input sentence are used to train the model. Therefore, pretraining a model using MLM requires considerable computational resources. Additionally, the token [MASK], which represents the mask, exists only during pretraining and does not appear during fine-tuning. This mismatch of [MASK] tokens between pretraining and fine-tuning slightly degrade the performance of MLM pretraining models.

4.2 Replaced Token Detection

ELECTRA employs a new pretraining method called RTD to improve the weaknesses of MLM. It learns a bidirectional model like MLM while learning from all input sentences like a conventional language model.

Based on the idea of GAN, RTD trains the generative model to distinguish between “real” and “fake” input words. Rather than collapsing the input sentence by replacing a word with [MASK], as BERT dose, RTD collapses the input sentence by replacing it with a “fake” word that is false but plausible compared to the original sentence. Thereafter, the discriminative model is trained to take the
collapsed sentence as input and to predict which words have been replaced compared to the original input sentence.

The generator model can be any model that produces token output distributions, but a previous study (Clark et al., 2020) used a small MLM model trained concurrently with the discriminator model, i.e., the BERT model with a small hidden layer scale. Although the relationship between the generator and discriminator is similar to the structure of GAN described above, applying GAN to the text domain is not easy. Therefore, the generator is trained using the maximum likelihood to predict masked words rather than adversarial. If the generator correctly predicts the original word of the masked word, the word is labeled as the original word.

The generator and discriminator share the same input word embeddings, and after pretraining, the generator is removed and only the discriminator is adjusted in the downstream task. Both the two models use the neural transformer architecture.

### 4.3 Analyzing the Efficiency of MLM

MLM is considered inefficient, and to confirm this, Clark et al. (2020) set up and experimented with three models. The performance of ELECTRA is improved by defining the loss for all input tokens instead of only partially. BERT’s performance is slightly impaired by the discrepancy between pretraining and fine-tuning in the [MASK] token. BERT has already been replaced with random tokens as a measure to improve this mismatch; however, even this measure is not sufficient. Consequently, ELECTRA is computationally more efficient than BERT owing to more efficient tokens and less mismatch during fine-tuning (use of MASK symbols). The improvement in ELECTRA can also be attributed to other factors than fast learning. ELECTRA outperforms BERT by a wider margin when the model size is smaller, and it converges perfectly. Although more analysis is needed, ELECTRA is generally considered more parameter efficient than BERT.

Because ELECTRA-JP-Small and Tohoku-BERT model sizes are different, we did not confirm the exact difference in computational efficiency, but if the two Japanese models of the same size are trained together, ELECTRA will likely produce better accuracy.

### 4.4 Unsupervised Domain Adaptation

Models that generate contextualized word embeddings, such as ELMo and BERT, perform well across natural language processing tasks when pretrained on large unlabeled corpora. Although these models use corpora such as Wikipedia or news texts for training, it is unclear whether this approach is effective when the domain and writing style of the target text differ significantly from the pretrained corpus. However, fine-tuning the distributed representation using the text in the target domain improved the performance (Han and Eisenstein, 2019).

In the related research, two texts were tested as target domains: Early Modern English and Twitter. Both are different from the existing pretrained corpora, but the proposed method provides substantial improvements over the BERT model.

We use the computationally efficient ELECTRA model for fine-tuning the target domain instead of BERT to improve the accuracy in downstream tasks and increase the computational speed using a smaller model.

### 4.5 Domain/Task Tuning

Contextualized word embeddings may be ineffective for tagging tasks when the target domain is different from the pretrained corpus. This is especially serious for unsupervised domain adaptation because the labeled data may differ significantly from the target text. To address this problem, a method of the AdaptaBERT model for unsupervised domain adaptation was proposed.

Specifically, the following two approaches have been applied.

**Domain Tuning** Unsupervised tuning of the BERT language model using text from the target domain. For example, BERT trained on Wikipedia can be retrained using Twitter text.

**Task Tuning** A way to tune BERT using teacher data. For example, for named entity recognition, this would be to train the entire model including BERT using CoNLL 2003.

The four experimental settings are defined by the combination of domain and task tunings.

- Frozen BERT
- Task-tuned BERT
- AdaptaBERT
• Fine-tuned BERT

Frozen BERT is a method that uses BERT as a feature extractor without training it. Task-tuned BERT is a method that BERT is trained using supervised data from the source domain. AdaptaBERT is a method in which BERT is tuned using unsupervised data from the target domain and then the model is trained using supervised data from the source domain. The last one, Fine-tuned BERT, is a method to train the entire model including BERT using data from the target domain.

The experiment evaluates part-of-speech tagging in the Penn Parsed Corpus of Early Modern English (PPCEME) and named entity recognition in the Workshop on Noisy User Text (WNUT) 2016.

For part-of-speech tagging, used the Penn Treebank (PTB) corpus of 20th century English as the source domain corpus and PPCEME as the target domain corpus. PTB corpus is for modern English, and the PPCEME corpus is for 15th to 17th century English.

For named entity recognition, used Conference on Natural Language Learning (CoNLL) 2003 as the source domain corpus and WNUT 2016 as the target domain corpus. CoNLL 2003 is for news, and WNUT is for Twitter.

The results of part-of-speech tagging and named entity recognition show that using Domain Tuning to train BERT on a corpus of the target domain improves performance. Even if no large amount of labeled data in the target domain is unavailable, improving performance by simply tuning the pretraining model using unsupervised data from the target domain is practical.

In our experiments, we assume a situation where a large amount of labeled data in the target domain cannot be secured. We are exploring how much Domain Tuning can improve the performance of ELECTRA-Small under this situation.

5 Conclusion

In this paper, focusing on the computational efficiency of the ELECTRA model and its good performance at scale, we constructed a domain-specific pretrained ELECTRA model by additional pretraining it using a small corpus of the target domain.

Although the constructed ELECTRA model is smaller than the Tohoku-BERT model to be compared, it achieved higher performance than Tohoku-BERT in document classification in the target domain.

In future work, we would like to construct a pretraining model of the same size to compare the performance of the two models more rigorously.

Acknowledgment

The research was supported by JSPS KAKENHI Grant Number JP19K12093 and ROIS NII Open Collaborative Research 21FC05.

References


BERT-PersNER: a New Model for Persian Named Entity Recognition

Farane Jalali Farahani
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
fjalali@ce.sharif.edu

Gholamreza Ghassem-Sani
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
sani@sharif.edu

Abstract

Named entity recognition (NER) is one of the major tasks in natural language processing. A named entity is often a word or expression that bears a valuable piece of information, which can be effectively employed by some major NLP tasks such as machine translation, question answering, and text summarization. In this paper, we introduce a new model called BERT-PersNER (BERT based Persian Named Entity Recognizer), in which we have applied transfer learning and active learning approaches to NER in Persian, which is regarded as a low-resource language. Like many others, we have used Conditional Random Field for tag decoding in our proposed architecture. BERT-PersNER has outperformed two available studies in Persian NER, in most cases of our experiments using the supervised learning approach on two Persian datasets called Arman and Peyma. Besides, as the very first effort to try active learning in the Persian NER, using only 30% of Arman and 20% of Peyma, we respectively achieved 92.15%, and 92.41% performance of the mentioned supervised learning experiments.

1 Introduction

Named Entity Recognition (NER) is a fundamental Natural Language Processing (NLP) task, in which we try to identify and classify certain entities such as organizations, persons, locations, etc. in a given text. A named entity is often a word or expression in the text that bears a valuable piece of information, which can be effectively used in other high-level NLP tasks such as machine translation, question answering, and text summarization. Unlike English, which is rich in digital resources, there are some natural languages such as Persian, which is regarded to be a low-resource language. However, Persian is an important language because it is spoken by more than 110 million people worldwide.

In this article, we introduce an efficient model for Persian NER. To tackle problems such as the lack of labeled data in Persian, we have used two different learning methods, i.e., transfer learning (Pan and Yang, 2010) and active learning (Settles, 2010). To transfer knowledge in transfer learning, the model-based (parameter) approach (Settles, 2010) has been used to fine-tune BERT (Devlin et al., 2019). Through transfer learning, we can produce a powerful model using limited data that takes much less training time. We also employed active learning in order to perform the fine-tuning task by using only a few informative samples instead of the whole dataset. Through active learning, it is possible to deliver a performance that is very close to that of a supervised learning method.

The structure of this paper is as follows. We review the literature in Sec. 2; BERT is discussed in Sec. 3; Our proposed method is explained in Sec. 4; The results of our experiments are given in Sec. 5 and the article is concluded in Sec. 6.

2 Literature Review

2.1 Named Entity Recognition

The models that have tackled NER through Deep Learning (DL) consist of three main components: 1) input representation, 2) context encoder, and 3) tag decoder (Li et al., 2020). For the first component, Word2Vec (Mikolov et al., 2013) was used in (Lample et al., 2016) and both Word2Vec and GloVe (Pennington et al., 2014) were used in (Poost-
chi et al., 2018). As the second component, in most studies, a bidirectional long short-term memory (BiLSTM) has been used to capture long-distance dependencies (Li et al., 2020; Shahshahani et al., 2019; Lample et al., 2016; Bokaei and Mahmoudi, 2018; Poostchi et al., 2018). In other studies, an architecture named Transformer (Vaswani et al., 2017) was used. In the architecture of Transformer, there is no recurrent structure and it operates based on attention mechanisms, which lead to an increase in parallelization (Vaswani et al., 2017). The Pre-trained BERT model is based on the Transformer architecture and supports more than 100 live languages, including Persian. It has also been applied to NER (Devlin et al., 2019; Taher et al., 2019). As the last component, Conditional Random Field (CRF) (Lafferty et al., 2001) has been mostly used as the referred tag decoder. The main reason for choosing CRF is that, instead of merely looking for the best label (tag) for each word, it jointly uses neighboring tags to determine a sequence of output labels (Li et al., 2020; Lample et al., 2016; Bokaei and Mahmoudi, 2018; Poostchi et al., 2018; Taher et al., 2019).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Entity type</th>
<th>#Tokens</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arman</td>
<td>Person</td>
<td>5215</td>
<td>2.08</td>
</tr>
<tr>
<td></td>
<td>Organization</td>
<td>10036</td>
<td>4.01</td>
</tr>
<tr>
<td></td>
<td>Location</td>
<td>4308</td>
<td>1.72</td>
</tr>
<tr>
<td></td>
<td>Facility</td>
<td>1485</td>
<td>0.59</td>
</tr>
<tr>
<td></td>
<td>Product</td>
<td>1463</td>
<td>0.58</td>
</tr>
<tr>
<td></td>
<td>Event</td>
<td>2518</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Other</td>
<td>224990</td>
<td>89.99</td>
</tr>
<tr>
<td>Peyma</td>
<td>Person</td>
<td>7675</td>
<td>2.33</td>
</tr>
<tr>
<td></td>
<td>Organization</td>
<td>16964</td>
<td>5.6</td>
</tr>
<tr>
<td></td>
<td>Location</td>
<td>8782</td>
<td>2.90</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>732</td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td>Date</td>
<td>4259</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>Money</td>
<td>2037</td>
<td>0.67</td>
</tr>
<tr>
<td></td>
<td>Percent</td>
<td>699</td>
<td>0.23</td>
</tr>
<tr>
<td></td>
<td>Other</td>
<td>261382</td>
<td>86.39</td>
</tr>
</tbody>
</table>

Table 1: Details of Arman and Peyma datasets (Poostchi et al., 2018; Shahshahani et al., 2019).

2.2 Dataset

One of the datasets that have been used in Persian NER is called Arman, which was first published in 2016. It consists of about 250k tokens and six classes: location, organization, person, facility, product, and event (Poostchi et al., 2018) 1. In 2018, another dataset called Peyma was published in (Shahshahani et al., 2019), which includes 300k tokens and seven classes: location, organization, person, time, data, money, and percent, Table 1 2.

2.3 Active Learning

The active learning strategies are divided into three groups: pool-based sampling, stream-based sampling, and membership query synthesis (Settles, 2010). In the first two, the instances are taken from a pool of data and a stream of data, respectively. However, in the last one, instances are generated. The main advantage of the pool-based sampling is that it provides the possibility of running a comparison among all instances and selecting the most informative samples for training the model. The pool-based sampling method has been applied to English NER in several studies (Shen et al., 2017; Chen et al., 2015).

In (Shen et al., 2017), OntoNotes-5.0 English and Chinese were used for active learning experiments. The authors employed different selection strategies, where in each case, an LSTM-based model was firstly trained with 1% of the original training dataset. Then, in each round, the most informative instances were selected from the remaining 99% for training; and each round was ended when 20,000 words had been added to the training dataset. The training process was repeated at the end of each round based on the accumulated dataset and the parameters of the model were updated through this repetition of training. They showed that by this active learning process, one could achieve 99% performance of supervised learning, using only 30.1% of the Chinese dataset. It was also shown that the same performance could be achieved using only 24.9% of the English dataset.

In (Chen et al., 2015), the NER task was handled for medical texts. The authors used

---

1Arman is available at: https://github.com/HaniehP/PersianNER/blob/master/ArmanPersoNERCorpus.zip
2Peyma is available via a folder named 300K at: http://en.itrc.ac.ir/sites/default/files/pictures/NER.rar
the 2010 i2b2/VA annotated dataset. They randomly split the dataset into two parts: (1) a pool including 80% of the data for being used during the active learning process and (2) a test set including the remaining 20% for assessing the NER model. The authors simulated practical pool-based active learning by getting labels from the mentioned pool instead of interacting with an actual user. It was emphasized that the labels had not been accessed unless the active learning algorithm selected an instance for being added to the training data. They gained F1 score of 80% on the 2010 i2b2/VA dataset, using only 58% of the original training data.

To the best of our knowledge, active learning has not been previously applied to Persian NER and our work is the first examination of this learning method in Persian NER.

3 BERT

As it was mentioned before, the architecture of BERT is based on Transformer. BERT is a pre-trained model, trained on about 3.3 billion unlabeled data. Its input embedding consists of three parts: 1) token embedding, 2) segment embedding, and 3) position embedding (Devlin et al., 2019). For token embedding, WordPiece has been employed. Segment embedding is used to distinguish a pair of input sentences. The goal of position embedding is to determine sentence word order as the words of each sentence are fed to BERT simultaneously and without any specific order (Devlin et al., 2019).

BERT is applicable in both freezing and fine-tuning methods (Pan and Yang, 2010). In freezing, no change is applied to the pre-trained model. Freezing leads to the extraction of constant features from the model so that they can be used as contextualized word embeddings. In contrast, in the fine-tuning method, the parameters of certain layers are fine-tuned based on a down-stream task and labeled data. It was shown that fine-tuning outperforms freezing (Peters et al., 2019). Therefore, we have used fine-tuning in our proposed method. Besides, like several previous studies, we have used CRF for the tag decoding. The detailed account of our proposed method, which we are going to refer to as BERT-PersNER, is given next.

4 Proposed Method

4.1 BERT-PersNER

BERT has been used as the first two sections of our NER architecture (i.e., the input representation and the context encoder sections). BERT generates an intermediate representation for each token. These intermediate representations will be later used to predict the output label sequences. As the tag decoder section of the mentioned NER architecture, we have used CRF for predicting output labels. We have used a number of fully connected neural networks to ensure that the BERT output vector dimension will match with the number of possible tags for each token. The architecture of BERT-PersNER is shown in Figure 1.

Assuming that $x = \{x_1, x_2, \cdots, x_N\}$ shows the observed input tokens of length N, and $y = \{y_1, y_2, \cdots, y_N\}$ represents the corresponding output labels in a linear-chain CRF, $P(y|x)$ is calculated as follows:

$$P(y|x) = \frac{e^{Score(x,y)}}{\sum_{y' \in Y(x)} e^{Score(x,y')}},$$ (1)
where
\[
Score(x, y) = \sum_{i=0}^{N} T_{y_i, y_{i+1}} + \sum_{i=1}^{N} P_{y_i} \quad (2)
\]
In equation 2, the transition matrix \( T \in R^{K+2 \times K+2} \), in which \( K \) is the count of distinguished labels and +2 is added for the beginning and ending labels, represents a transformation from one label to another. The transition matrix is initialized randomly and is updated during training. The fully connected neural network output is in the \( P \) matrix, where \( P_{i,y_i} \) represents the score of label \( y_i \) for \( i^{th} \) input token. \( Y(x) \) is the set of all possible labeled sequences for \( x \). The loss function equals the negative log-likelihood and the best sequence label obtained through the following equation and the Viterbi algorithm:
\[
y^* = \arg\max_{y' \in Y(x)} \log P(y'|x) \quad (3)
\]

4.2 Active Learning in BERT-PersNER

In this work, similar to some earlier researches (Shen et al., 2017; Chen et al., 2015), we have simulated the pool-based sampling method of active learning and, instead of interacting with an actual user for obtaining labels, an annotated dataset has been used. Akin to some earlier works, we randomly divided the original training dataset into two parts: an initial batch named \( L \), which contained 1% of the data, and a pool named \( U \), which included the remaining 99%. Note that the label of no instance in \( U \) was accessed unless that our selection strategy chose that instance for being added to \( L \).

The active learning framework in BERT-PersNER consists of the following four steps:

1. Building the initial model: to begin with, the initial BERT-PersNER model is obtained through performing the training process using the \( L \) batch.

2. Sorting the instances: at this stage, based on a selection strategy, the informativeness of the instances in \( U \) is measured (i.e., done without accessing the labels). Then, 10% of the top-ranked instances, together with their labels, are removed from \( U \) and added to \( L \).

3. Training: the BERT-PersNER model is then trained again on the new \( L \) and the parameters of the model are updated.

4. Iterating: stages 2 and 3 are repeated until that the instances of \( U \) are exhausted.

The selection strategy can be regarded as the most important part of active learning, because the informativeness of each instance is determined by this strategy. In this study, uncertainty sampling (Lewis and Gale, 1994) has been used as our selection strategy. This selection strategy is based on the idea that if the label of those instances on which the model is less certain is known to the system, it will be more beneficial. The following three uncertainty sampling strategies have been implemented in this study:

- Normalized Least Confidence (NLC) (Lewis and Gale, 1994): in this strategy, the certainty of the best label sequence for each input sample is used as a criterion to find the least confident instances. To eliminate the length effect of input sentences, for each instance, a Normalized form of least confidence is calculated through the following equation:
  \[
  \phi^{NLC}(x) = 1 - \frac{1}{N} P(y^*|x; \theta), \quad (4)
  \]
  in which \( x \) is the input sequence, \( y^* \) represents the best label sequence for \( x \), \( N \) is the length of \( x \), \( \theta \) shows the model parameters, and \( P \) is the instance confidence (i.e., based on equation 1).

- Margin (M) (Scheffer et al., 2001): in this strategy, using the following equation, the marginal difference between the first two best label sequences is used as the criterion for finding the least confident instances:
  \[
  \phi^{M}(x) = -(P(y_1^*|x; \theta) - P(y_2^*|x; \theta)), \quad (5)
  \]
  in which the negative sign is added to select the instances with the lowest margin.

- Sequence Entropy (SE) (Settles and Craven, 2008): here, the entropy of all label sequences is used as the criterion to
find the least confident instances through the following equation:

$$
\phi_{SE}(x) = - \sum_{y'} P(y'|x; \theta) \log P(y'|x; \theta)
$$

Since the number of possible label sequences grows exponentially as our input sentence length increases, we have only considered the N-best (with N = 16) label sequences for each instance.

### 5 Experiments

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Entities</th>
<th>Word-level</th>
<th>Phrase-level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arman</td>
<td>B-Person</td>
<td>92.26</td>
<td>93.59</td>
</tr>
<tr>
<td></td>
<td>B-Organization</td>
<td>81.61</td>
<td>87.97</td>
</tr>
<tr>
<td></td>
<td>B-Location</td>
<td>82.08</td>
<td>78.67</td>
</tr>
<tr>
<td></td>
<td>B-Facility</td>
<td>75.62</td>
<td>80.78</td>
</tr>
<tr>
<td></td>
<td>B-Product</td>
<td>70.95</td>
<td>75.30</td>
</tr>
<tr>
<td></td>
<td>B-Event</td>
<td>70.95</td>
<td>78.83</td>
</tr>
<tr>
<td></td>
<td>All classes</td>
<td>84.23</td>
<td>80.80</td>
</tr>
<tr>
<td>Peyma</td>
<td>B-Location</td>
<td>86.78</td>
<td>76.02</td>
</tr>
<tr>
<td></td>
<td>B-Person</td>
<td>86.88</td>
<td>91.19</td>
</tr>
<tr>
<td></td>
<td>B-Organization</td>
<td>83.09</td>
<td>87.23</td>
</tr>
<tr>
<td></td>
<td>B-Time</td>
<td>75.90</td>
<td>82.72</td>
</tr>
<tr>
<td></td>
<td>B-Date</td>
<td>84.23</td>
<td>86.91</td>
</tr>
<tr>
<td></td>
<td>B-Money</td>
<td>92.52</td>
<td>92.01</td>
</tr>
<tr>
<td></td>
<td>B-Percent</td>
<td>91.64</td>
<td>94.48</td>
</tr>
<tr>
<td></td>
<td>All classes</td>
<td>86.14</td>
<td>82.05</td>
</tr>
</tbody>
</table>

Table 2: F1 scores (in percentage) of running our model on Arman and Peyma.

Word-level evaluation of BERT-PersNER on Arman shows that the best performance of this model is achieved on I-person, and its weakest performance is where it deals with B-event and B-product. On the other hand, in the phase-level evaluation, the model performs best on Person and worst on Event. This is mainly due to the differences between tag counts in Arman; as it is depicted in Table 1, the frequency of Person tags is twice as many as Event tags, and four times as that of Product. Note that although Location is the most frequent tag in Arman, its resemblance to Organization causes that the BERT-PersNER performance on Location to become weaker than its performance on Person.

Word-level evaluation of BERT-PersNER on Peyma shows that its best performance is on I-percent and its worst performance is on B-time. On the other hand, phrase-level evaluation of the model indicates that it works best on Percent and its weakest performance is on Time. This is due to the fact that, although the tag count of Percent in Peyma is not high, the low level of variety between different Percent tags allows the model to learn this class effectively. On the other hand, due to a low tag count of Time and resemblance to Date make BERT-PersNER acts worst on Time.
Table 4: A comparison between F1 scores (in percentage) of different selection strategies in active learning.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Selection Strategy</th>
<th>Percent of training data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Arman</td>
<td>NLC</td>
<td>70.40</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>68.91</td>
</tr>
<tr>
<td></td>
<td>SE</td>
<td>71.84</td>
</tr>
<tr>
<td></td>
<td>RAND</td>
<td>65.89</td>
</tr>
<tr>
<td>Peyma</td>
<td>NLC</td>
<td>75.60</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>72.68</td>
</tr>
<tr>
<td></td>
<td>SE</td>
<td>72.46</td>
</tr>
<tr>
<td></td>
<td>RAND</td>
<td>72.26</td>
</tr>
</tbody>
</table>

Figure 2: BERT-PersNER performance on Arman, using different selection strategies

Figure 3: BERT-PersNER performance on Peyma, using different selection strategies

Table 3 shows BERT-PersNER performance against two recent studies (Shahshahani et al., 2019; Bokaei and Mahmoudi, 2018). As it is observed, our proposed model outperforms the baselines on Arman. More precisely, word-level evaluation, BERT-PersNER improved Deep-CRF by 2.73%. Furthermore, in phrase-level evaluation, it enhanced the results of the baseline by 4.01%.

As it is shown in 3, a word-level evaluation of BERT-PersNER performance on Peyma against that of (Shahshahani et al., 2019) did not result in any improvement. However, in a phrase-level evaluation, it has improved the result of baseline by 2.05% of F1 score.

In all our active learning experiments, the evaluation of BERT-PersNER has been performed in word-level. Since a gradual increase of data on its own can improve the model performance, in addition to the previously mentioned selection strategies, we have used a random selection strategy as our baseline. Here, too, a three-fold cross validation has been employed and the final results have been averaged over the three-fold. In each iteration, the chosen selection strategy takes 10% of the remaining unlabeled data from the pool and adds it to the training dataset. That is about 507 and 546 sentences of Arman and Peyma, respectively.

As it can be observed in Figures 2 and 3, reflecting the results of Table 4, NLC has outperformed its counterparts. We think this is due to the fact that it elects the instances based on their very best label sequence, whereas other strategies also take other slightly weaker instances into consideration (e.g., the second best label sequence is used in Margin). As it is shown in Figure 2, on average, the performance of SE, M, and RAND on Arman has been respectively weaker than that of RAND. On Peyma (cf. Figure 3), again, RAND has been the weakest of all; but there has not been any clear superior between SE or M.
As it can also be seen in the mentioned figures, in the earlier stages of active learning, the gradient is much higher than its later stages, which is an indication of selecting much more informative instances in those earlier stages.

Using only 30% of Arman, NLC achieved 92.15% performance of supervised learning. On the other hand, in the case of Peyma, using 20% of data, NLC reached 92.41% performance of the supervised learning approach. Therefore, by using more informative instances, we can reach a performance compatible with that of supervised learning with much less required data. This data saving is particularly critical in the case of low-resource languages such as Persian.

6 Conclusion

We fine-tuned the pre-trained BERT model for the task of Named Entity Recognition in Persian, which is regarded as a low-resource language. We employed both transfer learning and active learning methods to develop a new model called BERT-PersNER for Persian NER. The new model was evaluated on two Persian NER datasets, which are called Arman and Peyma. We first evaluated BERT-PersNER in a supervised approach, which was done on both word- and phrase-level. Our new model outperformed two previous studies in Persian NER by 2.05%, 2.73%, and 4.01% F1 scores, in phrase-level on Peyma, in word-level on Arman, and in phrase-level on Arman, respectively. In the word-level evaluation on Peyma, however, the performance of BERT-PersNER was lower than the best available related work by 0.86% F1 score.

BERT-PersNER was evaluated word-level using the active learning approach with four different selection strategies. As our main contribution, it was shown that using only 30% of Arman, we can achieve 92.15% performance of the supervised learning method. It was also shown that using only 20% of Peyma, we can reach 92.41% performance of the supervised learning case. To the best of our knowledge, the application of active learning in Persian NER is the very first effort in this respect. As our future work, we intend to investigate the impact of other selection strategies on BERT-PersNER. We also plan to evaluate the proposed approach using other newly published pre-trained models.

References


Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020. A survey on deep learning for named entity recognition. IEEE Transactions on Knowledge and Data Engineering.


Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors


Cross-lingual Fine-tuning for Abstractive Arabic Text Summarization

Mram Kahla and Zijian Győző Yang and Attila Novák
Pázmány Péter Catholic University
Faculty of Information Technology and Bionics
Práter u. 50/a, 1083 Budapest, Hungary
{lastname.firstname (.midname)}@itk.ppke.hu

Abstract

While abstractive summarization in certain languages, like English, has already reached fairly good results due to the availability of trend-setting resources, like the CNN/Daily Mail dataset, and considerable progress in generative neural models, progress in abstractive summarization for Arabic, the fifth most-spoken language globally, is still in baby shoes. While some resources for extractive summarization have been available for some time, in this paper, we present the first corpus of human-written abstractive news summaries in Arabic, hoping to lay the foundation of this line of research for this important language. The dataset consists of more than 21 thousand items. We used this dataset to train a set of neural abstractive summarization systems for Arabic by fine-tuning pre-trained language models such as multilingual BERT, AraBERT, and multilingual BART-50. As the Arabic dataset is much smaller than e.g. the CNN/Daily Mail dataset, we also applied cross-lingual knowledge transfer to significantly improve the performance of our baseline systems. The setups included two M-BERT-based summarization models originally trained for Hungarian/English and a similar system based on M-BART-50 originally trained for Russian that were further fine-tuned for Arabic. Evaluation of the models was performed in terms of ROUGE, and a manual evaluation of fluency and adequacy of the models was also performed.

1 Introduction

When we talk about text summarization that practically means that using certain algorithms, we teach a machine to subtract information from an extensive text and provide a significantly shorter overview of it. And just like in the case of human beings, like a number of cases in our own school experiences, there are two ways of doing that.

The first way is called extractive summarization (Nallapati et al., 2016; Zhang et al., 2018; Narayan et al., 2018). In this method the idea is to practically highlight, take out certain keywords, phrases, or sentences from the text and put them together. Therefore, the result, the summary will use the exact same words, terms, and sentences as the original, and almost certainly even in the same order. It is practically the same method, when we humans glance through a massive text, like a thick script, or a book in just a few minutes. The machine would, just like our mind, focus on the first few words, paragraph, or page, then pick up the most commonly occurring words, and extract complete sentences with them, without changing anything in the sentence.

The second way is abstractive summarization (See et al., 2017; Paulus et al., 2017; Rush et al., 2015). Once again, this is not something new for our brain. Like in so many of our school studies, when a certain assignment was given to us and then later asked about in school, we probably did not use the same words and phrases of the original reading material, but we had a general idea about it, which we explained in our own plain words. And since our limited capability to remember and the inherent tendency for laziness, our summary was usually short, only an essence reflecting the original passage. Quintessentially that is abstractive summarization. Creative reconstruction of a textual message built on its comprehension.

We chose to focus on the Arabic language because it presents us challenges, which once surpassed can open up new fields of research.

Arabic has many features. One of the advantages of the Arabic language is that besides its huge variety of dialects and spoken versions - which is a naturally occurring phenomenon - its formal written version (Fusha), a practically dead language, is highly standardized and lacks major regional varia-
ety. But it still has a massive amount of primary users - native Arabic speakers - and text providers, also a big amount of secondary users, meaning people who are not native speakers themselves but in their engagements with Arab people, or organizations create new texts. This sort of lingual stability is rare among "big languages". So that would be the good side that we have a fairly standard massive corpus with minimal presence of dialectical variables complicating the learning process. On the other hand, Arabic is a language in which short vowels are not written. Though possible, usually not even marked in texts, therefore it is required for the reader to have extensive knowledge of the language, otherwise, the reader would not comprehend the message of the letters, even if he/she knows them.

For better comprehension let us see an example. In English, a given word, like "wish" is always written and read the same way. It might be understood as a verb, or an identically written noun, so there is a number of variables we can attach to the word but given the surrounding text that is easy to process. With some words, like "read", which can be either past or present, and the application of idioms, this number of variables grows, but not significantly. And the number of words presenting such a feature is also fairly limited, as these are rather highlightable exceptions than rules. In Arabic, however, a three-letter word k-t-b, can automatically present 3 distinct forms, namely "he wrote", "it was written" and "books". With minimal alteration, the number of possible solutions goes up to 20, or so, and that is a base rule with very few exceptions. And here we are still not talking about an agglutinative language, like Hungarian and Turkish, where the suffixes with a big number of variables, but cutting them the core stays fairly the same. With Arabic, we face a massive amount of inherent variables, all to be taken into consideration upon processing. The problem, however, presents an opportunity, as we can exploit this phenomenon to achieve bigger coherence in the text digestion.

Arabic also has another advantage, both scientifically and in the sense of application. It is one of the only 6 U.N. official languages, along with English, French, Spanish, Russian, and Chinese. That means that we have a massive resource of scientific and checked texts, beyond the usual quantity, upon which can be built, and which can be a potential test ground for further development. Meaning that we not only have a large quantity of informal, or semiformal text between native speakers, but we also have a huge reviewed linguistically double-checked text. Given it is a U.N. language, massive amount of texts, which otherwise would have necessarily not concerned the Arab world, are translated into it and linguistically checked by professionals. With the inclusion of the political and economic value of the region, and the amount of politically sensitive and important material to be assessed, the value of a text summarization tool for Arabic cannot be overstated.

The main contributions presented in this paper include a) presenting the first corpus for abstractive Arabic text summarization, b) several neural models to perform abstractive news summarization for Arabic, and c) evaluation of the performance of these models. In addition to leveraging linguistic knowledge embodied in pretrained neural language models (using multilingual BERT, a transformer encoder trained on 104 Wikipedia languages including Arabic, and AraBERT, a monolingual BERT model trained specifically for Arabic), we also apply cross-lingual transfer to improve our results. We use summarization models based on multilingual neural language models (multilingual BERT and multilingual BART-50, a pre-trained sequence-to-sequence model for 50 languages including Arabic) that were originally fine-tuned to do summarization in another language, and further fine-tune them for the Arabic summarization task. We thus also leverage the knowledge of the original models concerning the summarization task they learned from resources in other languages.

The rest of the paper is structured as follows. Section 2 presents related work published on Arabic summarization. Section 3 describes the methodology and the experiments that we have done. Section 4 describes the results of automatic and manual evaluation. The last section concludes the paper.

2 Related Work

The work for Arabic summarization is limited. Most existing systems use the extractive approach. Lakhas (Douzidia and Lapalme, 2004) is considered the first extractive Arabic summarization system that was evaluated and compared with systems processing English input. The system produces a 10-word summary and translates it to English and then it is evaluated using the ROUGE measure.
Another Arabic text summarization approach based on fuzzy logic was proposed by Qassem et al. (2019). This model is based on a new noun extraction method and fuzzy logic. Yet another Arabic text summarization tool, SumSAT, (Lakhdar and Chéragui, 2019) adopts an extractive approach using a hybrid of three techniques: a) contextual exploration which allows access to the semantic content of a text, without the need for deep syntactic analysis; b) identification of indicative expressions offering the possibility of generating a summary in a general topic or a specific domain by selecting sentences that contain specific indicators, and c) the graph method which generates the summary by selecting the most representative phrases of the source text.

The evaluation process differs between these systems, as well as the datasets used for evaluation. Lakhas (the first extractive Arabic summarization system) was evaluated cross-lingually. It generates a summary, translates it to English, and then it evaluates the English summary using the ROUGE-N measure. For that aim 240 documents and their corresponding summaries were produced and used as a dataset. In the case of the SumSAT tool, performance was evaluated in terms of precision and recall of the discursive annotation generated by SumSAT and manual reference annotations (i.e. it was not summaries per se that were evaluated). For the evaluation, they constructed a dataset composed of 25 documents and their corresponding annotation. In contrast with the above, (Al Qassem et al., 2019) evaluated the summaries using ROUGE-N (N=1 and 2) metric and evaluated the summarizer using the Essex Arabic Summaries Corpus (EASC) (El-Haj et al., 2010), which contains 153 Arabic articles and 765 human-generated extractive summaries of those articles created using Mechanical Turk.

An RST-based\(^1\) automatic summarization technique for Arabic texts was presented by (Maaloul et al., 2010) which was implemented through the ARSTResume system. They created a corpus of Arabic texts from a newspaper website, and they claimed that the ARSTResume evaluation showed encouraging results based on 50 texts.

Some recently published work also aims to address abstractive Arabic text summarization. Azmi and Altmami (2018) proposed a four-phase abstractive summarizer for Arabic where the core of the system is an extractive summarizer. The four phases are topic segmentation, headline generation, extractive summarization, and sentence reduction. For evaluation, they conducted two experiments. The first is to evaluate the extractive summarizer. For that they used 32 sample documents from two popular Saudi newspapers. The second is to evaluate the abstractive summarizer. For that they used 150 documents from six different Arabic newspapers. In addition, two linguist experts judged the quality of the abstractive summaries.

Another system (Al-Maleh and Desouki, 2020) was trained to generate headlines based on the first paragraph of Arabic articles, a task that can be classified as a kind of abstractive summarization. The authors used a sequence-to-sequence model implementing the pointer-generator approach including a copy mechanism as presented in See et al. (2017). For training and evaluation, they crawled an Arabic data-set consisting of approximately 300 thousand article headline : introductory paragraph pairs.

An attempt at abstractive Arabic text summarization proper was presented in (Elmadani et al., 2020) applying multilingual-BERT-based (Devlin et al., 2019) models for both abstractive and extractive summarization using the models presented in (Liu and Lapata, 2019) trained and tested on the KALIMAT dataset (El-Haj and Koulali, 2013). A shortcoming of the research is, however, that the 20,291 article summaries in KALIMAT are machine-generated summaries output by the extractive Gen-Summ (=AQBTSS) algorithm (El-Haj et al., 2010). The train/test sets are thus neither human generated nor abstractive. Both studies evaluated the summaries using the ROUGE metric (Lin, 2004).

3 Methodology

This paper reflects on a specific approach of abstractive text summarization applied to Arabic. In terms of model architecture, we focus on approaches based on now-ubiquitous large-scale pre-trained language models (LM), such as BERT (Devlin et al., 2019) and BART (Lewis et al., 2020), which obtained new state-of-the-art results in diverse natural language processing tasks, including text summarization. An important feature of BERT and BART is that both of them have a multilingual model available, M-BERT (Devlin et al., 2019) and M-BART:50 (Tang et al., 2020) that include Arabic among the languages supported. In addition, we
need a big enough training and evaluation dataset consisting of Arabic texts and their abstractive summaries. So, the first step we took was to compile a reliable Arabic abstractive news summary corpus.

3.1 Data Collection

While we could mention two recent papers attempting at something that could be categorized as abstractive summarization in Section 2, one of them dealt with headline generation instead of summarization proper, and the other used a machine-generated extractive summaries dataset for training and evaluation. The main bottleneck hindering progress in Arabic abstractive summarization is thus the lack of a sizable dataset. We first needed to overcome this problem. We needed to build an Arabic abstractive summarization corpus. A great source of such a resource could be the press, like in the case of the trend-setting CNN/Daily Mail dataset (See et al., 2017), as many news articles have a lead, a brief overview of the content spread out in the article, with details only supporting, but not altering the original message. The only problem is that news articles with reliable abstractive leads are difficult to find. It is quite often the case that the lead is just a copy of the first paragraph or contains clickbait content rather than containing a good abstractive summary.

Spending considerable effort on evaluating a wide range of sites from the Arabic versions of CNN, BBC, France 24, DW, Sky News to the most popular fully Arabic sites like al-Mayadeen, al-Âlam, al-Ahrâm, al-Akhbar, and Sada-elbalad, we identified two Arabic news resources that could be the basis of a good Arabic abstractive news summaries dataset: the Arabic version of the Deutsche Welle (DW) news website, which seems to be the one containing the best abstractive summaries and the Files section of Sada-elbalad. The latter resource from Sada-elbalad later turned out to contain many problematic items containing several diverse topics only some of which were mentioned in the summary, we thus dropped this resource.

We downloaded Arabic Deutsche Welle resources from Common Crawl. We only kept articles from the “Main/Top Stories” section, and filtered out all articles where either the main article text or the lead was too short or missing and items where the text is shorter than 4 times the length of the lead. The dataset that we used in the experiments consists of 21508 articles and their corresponding leads.

We performed data processing steps on the raw material (the collected articles) to be ready for subsequent processing. Data processing means a number of steps that naturally all differ significantly from one NLP task to another. While that is a sensitive process on its own, we also face another difficulty making it somewhat challenging to rely on the experiences of the already developed model. That is the peculiar nature of each language and that not much similar work has been done on Arabic, which has its own difficulties both as language and script.

We use Python since it is capable to handle the Arabic language. We also use NLTK platform since it is an appropriate tool for Arabic NLP and can be used for preprocessing text for text summarization task with Arabic. Based on our corpora we needed to perform text tokenization. Table 1 displays the main characteristics of the corpora.

3.2 Experiments

The aim of the main task of our work is practically to fine-tune pre-trained language models for our task which is abstractive Arabic text summarization. For this aim, we fine-tuned multilingual BERT (having Arabic among the languages covered) for abstractive Arabic text summarization using our own corpus.

We also fine-tuned AraBERT (Antoun et al., 2020) for abstractive Arabic text summarization using the same corpus. AraBERT is the result of pre-training a BERT model specifically for the Arabic language.

In addition, we propose a cross-lingual-transfer-based approach to improve our results. Using pre-trained multilingual BERT, we fine-tuned multilingual BERT for abstractive Hungarian text summarization using the HVG corpus (Yang et al., 2021) where the articles and corresponding leads were taken from a daily online newspaper. We further fine-tuned this model for abstractive Arabic text summarization using our own corpus.

We followed the same approach using English training data instead of Hungarian. We used the CNN/DailyMail summaries corpus containing over 300k unique news articles to first train an English summarization system fine-tuning multilin-
Table 1: Main characteristics of the corpus

<table>
<thead>
<tr>
<th></th>
<th>Articles</th>
<th>Leads</th>
</tr>
</thead>
<tbody>
<tr>
<td>segments</td>
<td>21,508</td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>19,807</td>
<td></td>
</tr>
<tr>
<td>test</td>
<td>1,701</td>
<td></td>
</tr>
<tr>
<td>token #</td>
<td>6,929,974</td>
<td>2,867,754</td>
</tr>
<tr>
<td>type #</td>
<td>290,138</td>
<td>178,614</td>
</tr>
<tr>
<td>avg sent #</td>
<td>14.420</td>
<td>1.469</td>
</tr>
<tr>
<td>avg sent # (median)</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>avg token #</td>
<td>412.052</td>
<td>35.131</td>
</tr>
<tr>
<td>avg token # (median)</td>
<td>279</td>
<td>37</td>
</tr>
<tr>
<td>avg token # (mBERT)</td>
<td>848.821</td>
<td>73.559</td>
</tr>
<tr>
<td>avg token # (mBERT, median)</td>
<td>573</td>
<td>79</td>
</tr>
<tr>
<td>avg token # (araBERT)</td>
<td>481.219</td>
<td>39.631</td>
</tr>
<tr>
<td>avg token # (araBERT, median)</td>
<td>326</td>
<td>42</td>
</tr>
<tr>
<td>avg token # (mBART)</td>
<td>664.582</td>
<td>57.549</td>
</tr>
<tr>
<td>avg token # (mBART, median)</td>
<td>448</td>
<td>62</td>
</tr>
</tbody>
</table>

Table 1: Main characteristics of the corpus

gual BERT. Then we further fine-tuned this model for Arabic on our corpus.

We also fine-tuned the multilingual BART-50 model, which supports 50 languages including Arabic, using our own corpus. Following the approach mentioned above, we used a model fine-tuned from M-BART-50 for abstractive Russian text summarization using the Gazeta corpus (Gusev, 2020). We further fine-tuned this model for abstractive Arabic text summarization using our own corpus. Table 2 displays the ROUGE results of Hungarian and English m-BERT fine-tuning, and Russian m-BART-50 fine-tuning.

4 Results

Measuring the performance of a summarization system can be done through either automatic or manual evaluation. We evaluated our experiments using the ROUGE automatic metric and compared them to other abstractive Arabic summarization results. We also evaluated our results manually since the reliability of automatic metrics is often perceived as insufficient.

4.1 Automatic Evaluation

Automatic evaluation metrics are the most widely used tools in the overwhelming majority of the research papers on the subject of summarization. We have evaluated our experiments using ROUGE (Lin, 2004). ROUGE-1 and ROUGE-2 measure overlap of word uni-grams and bi-grams respectively. ROUGE-L measures overlap of the longest common sub-sequence between two texts. When comparing the performance of the models that we trained using our relatively small Arabic corpus, we found that using an abstractive summarization model based on multilingual BERT already fine-tuned for English on the CNN/DailyMail dataset as a starting point to train an Arabic summarization model leads to huge improvements in performance, as shown in Table 3.

Table 2: ROUGE recall results of Hungarian m-BERT, English m-BERT, and Russian m-BART fine-tuning

<table>
<thead>
<tr>
<th>Model</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>mBERT Hun</td>
<td>47.02</td>
<td>19.72</td>
<td>39.29</td>
</tr>
<tr>
<td>mBERT Eng</td>
<td>60.32</td>
<td>25.79</td>
<td>56.91</td>
</tr>
<tr>
<td>mBART Rus</td>
<td>32.1</td>
<td>14.2</td>
<td>25.7</td>
</tr>
</tbody>
</table>

Table 3: ROUGE recall results of abstractive summarization

<table>
<thead>
<tr>
<th>Model</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>AraBERT</td>
<td>6.121</td>
<td>0.117</td>
<td>6.121</td>
</tr>
<tr>
<td>mBERT</td>
<td>5.134</td>
<td>0.186</td>
<td>5.134</td>
</tr>
<tr>
<td>mBERT+HUN</td>
<td>6.466</td>
<td>0.261</td>
<td>6.462</td>
</tr>
<tr>
<td>mBERT+ENG</td>
<td>16.363</td>
<td>2.524</td>
<td>16.363</td>
</tr>
<tr>
<td>mBART-50</td>
<td>6.817</td>
<td>0.382</td>
<td>6.809</td>
</tr>
<tr>
<td>mBART-50-rus</td>
<td>7.116</td>
<td>0.499</td>
<td>7.045</td>
</tr>
</tbody>
</table>

Note that although there is a significant difference between the measured performance, the numbers cannot be directly compared, because performance was measured on different test sets. Moreover, as it was mentioned in Section 2, TRANS.ABS was evaluated on KALIMAT, in
which summaries are neither human-generated nor abstractive, so that corpus is not in fact suitable for the evaluation of abstractive summarization systems.

4.2 Manual Evaluation

In spite of the recent rapid progress in the development of summarization models, standard automatic evaluation metrics have not developed for nearly 20 years. In our experiments, ROUGE scores determine that our proposed method ranked significantly better than the existing systems, but the ROUGE scores did not reflect the real quality of the summaries generated. For the sake of a more accurate assessment, we decided to conduct a human evaluation. We manually evaluated the summaries generated by the different models. In order to achieve this, we created a web-based evaluation platform containing 100 random samples. For each of the 100 sample articles, the platform displays the following:

- Article text: the article text.
- Lead: the article corresponding lead.
- mBART-50: results generated from fine-tuning M-BART-50 with our corpus.
- mBART-50-ru-gazeta: results generated from fine-tuning the already fine-tuned M-BART-50 for Russian to Arabic.
- BERT multilingual cased trained from English model: results generated from fine-tuning the already fine-tuned M-BERT for English to Arabic.
- BERT multilingual cased trained from Hungarian model: results generated from fine-tuning the already fine-tuned M-BERT for Hungarian to Arabic.
- AraBERT: result generated from fine-tuning AraBERT with our corpus.

The evaluation process was done by 3 human annotators, who are from different backgrounds and have different views. One (S) is from Syria, which is a Levant country, and Arabic is the annotator’s mother tongue. The second (M) is from Morocco (Northwest Africa), where another dialect is used, and we can’t say that Arabic is their spoken mother tongue. The third (H) is from Hungary, who is not a native speaker, but a professional translator. This variety of annotators, who all have different points of view and different approaches to the Arabic language, raises the evaluation standard and ensures more reliable results. Though the two native speakers are both proficient in Fusha, the minor regional stylistic differences and the difference in whether they rely on it as a primary or secondary language, give a different angle of evaluation. The Hungarian annotator, on the other hand, gives an outer, more “neutral” look to the annotation.

We conducted manual evaluation in two steps. The first step is "Ranking", we asked the annotators to evaluate the output of the models and assign marks to each summary from 1 to 6 as shown in Table 5.

<table>
<thead>
<tr>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: BEST</td>
</tr>
<tr>
<td>2: Very good</td>
</tr>
<tr>
<td>3: Good</td>
</tr>
<tr>
<td>4: Acceptable</td>
</tr>
<tr>
<td>5: Poor</td>
</tr>
<tr>
<td>6: Very poor</td>
</tr>
</tbody>
</table>

Table 5: Ranking scores for manual evaluation.

Given the results of the first step of evaluation, we chose the best model and asked the annotators for the second step of evaluation which is giving quality scores, in the range 1 to 5, concerning adequacy (to what extent the output covers most relevant information in the text) and fluency (Table 6).

The ranking results showed that the AraBERT model is the weakest model, while the model based on multilingual BERT first trained for English summarization and then fine-tuned for Arabic (m-BERT English) is the best-performing model.

The manual evaluation showed that the six mod-
Table 6: Adequacy and Fluency scores for manual evaluation.

<table>
<thead>
<tr>
<th>Adequacy</th>
<th>Fluency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: none</td>
<td>1: incomprehensible</td>
</tr>
<tr>
<td>2: little meaning</td>
<td>2: dis-fluent Arabic</td>
</tr>
<tr>
<td>3: much meaning</td>
<td>3: non-native Arabic</td>
</tr>
<tr>
<td>4: most meaning</td>
<td>4: good Arabic</td>
</tr>
<tr>
<td>5: all meaning</td>
<td>5: flawless Arabic</td>
</tr>
</tbody>
</table>

The models differ considerably, though in several areas they are difficult to compare. Output from the m-BERT English model usually comes very close to the original lead. Clarity and language proficiency is rarely a problem.

The m-BERT-based model first fine-tuned for Hungarian (m-BERT Hungarian) also generates good summaries, but usually in a very different way. The wording, structural order, and grammatical tools have a tendency to differ, but in most cases, the meaning does not change. It is usual for this model to generate somewhat (about 10%) longer summaries, but added content is usually explanatory rather than just simple text addition. In other words, these additions give depth to the summary and structural coherence. However, comparison is difficult due to the significantly different expressions used. In other words, while the English-trained BERT model almost recreates the original lead, the Hungarian-trained one formulates the content in a different way.

Summary generated by the model simply fine-tuned from multilingual BERT without pre-training on summaries in another language correlate with those generated by the English-trained model, but they contain significantly more grammatical and contextual errors. Sometimes the message is just the opposite of that of the original article, sometimes the syntax falls apart. Yet there is a good number of promising summaries. It seems like a promising model still in development. It seems unfinished.

The AraBert-based model is far the weakest. It is clearly insufficient for practical usage. This model has a notorious tendency of distorting or reversing the meaning of the text, coming up with disturbingly wrong interpretations. There is a very high number of huge, sometimes hilarious grammatical mistakes not present in the output of any other model. Most problematic, however, is that this model generates far the longest summaries. Often the size is double of that the original lead or the output of the first model, yet this lengthy text does not add anything relevant to the summary. It simply bloats the summary, but does not add content. It seems like randomly selected and poorly sewn together sentences from the original text itself, but with great alterations of the meaning.

The models based on multilingual BART-50 and the m-BART-50-based model first fine-tuned for Russian have almost equally good results to the m-BERT English model, with remarkable text quality and fluency.

There seems to be little chance for improvement for the AraBert-based model, unlike the others, which are very promising. For some of the summaries it cannot be determined whether they were written by a human editor or are machine-generated. See Figure 1,2. Table 7 shows the Kappa(Cohen, 1960) values of inter-annotator agreement. We used 4 metrics to measure the inter-annotator agreement: Fleiss’s Kappa (Fleiss), Krippendorff’s alpha coefficient (Krippen), Scott’s pi (Scott), Average Pairwise Cohen’s Kappa (Cohen). The values of the inter-annotator agreement for the m-BERT English model are substantial.

![Figure 1: Ranking results](image1.png)

![Figure 2: M-BERT English Adequacy and Fluency manual evaluation results](image2.png)

**5 Conclusion**

In this paper, we introduced the first corpus for abstractive Arabic text summarization, which we compiled with our own effort. Based on this corpus,
we fine-tuned multilingual-BERT and multilingual-BART-based models for Arabic abstractive summarization. We also proposed a cross-lingual-knowledge-transfer-based approach. We applied this approach to improve summarization quality, further fine-tuning models first fine-tuned from multilingual BERT for Hungarian or English summarization to generate Arabic summaries, and applying the same training scenario to Russian using an M-BART-50-based model. The results of the ROUGE metric and manual evaluation showed that the proposed approach led to significant improvements in performance and achieved state-of-the-art results. In the future, we would like to extend our corpus and perform experiments with other models such as the PEGASUS model.

Acknowledgments
The authors of the paper wish to express the deepest gratitude to Professor Gábor Prószéký for his unrelenting support and to Dr. Dániel Sógor and Youssef Messaoudi for their great work in the manual evaluation process. This research was implemented with partial support provided by the National Excellence Programme 2018-1.2.1-NKP-00008: “Exploring the Mathematical Foundations of Artificial Intelligence”.

Table 7: Evaluation of inter-annotator agreement

<table>
<thead>
<tr>
<th></th>
<th>Fleiss</th>
<th>Krippen</th>
<th>Scott</th>
<th>Cohen</th>
</tr>
</thead>
<tbody>
<tr>
<td>mBERT-E</td>
<td>0.613</td>
<td>0.613</td>
<td>0.612</td>
<td>0.613</td>
</tr>
<tr>
<td>mBERT-H</td>
<td>0.285</td>
<td>0.273</td>
<td>0.273</td>
<td>0.298</td>
</tr>
<tr>
<td>mBERT</td>
<td>0.312</td>
<td>0.301</td>
<td>0.298</td>
<td>0.317</td>
</tr>
<tr>
<td>AraBERT</td>
<td>0.310</td>
<td>0.303</td>
<td>0.304</td>
<td>0.308</td>
</tr>
<tr>
<td>mBART</td>
<td>-0.016</td>
<td>-0.015</td>
<td>-0.019</td>
<td>-0.016</td>
</tr>
<tr>
<td>mBART-R</td>
<td>-0.009</td>
<td>-0.043</td>
<td>-0.047</td>
<td>-0.005</td>
</tr>
<tr>
<td>mBERT-Eng</td>
<td>0.228</td>
<td>0.229</td>
<td>0.226</td>
<td>0.232</td>
</tr>
<tr>
<td>AD FL</td>
<td>0.175</td>
<td>0.177</td>
<td>0.174</td>
<td>0.174</td>
</tr>
</tbody>
</table>

References


Mahmoud El-Haj, Udo Kruschwitz, and C. Fox. 2010. Using Mechanical Turk to create a corpus of Arabic summaries. In *Language Resources (LRs) and Human Language Technologies (HLT) for Semitic Languages workshop in conjunction with the 7th International Language Resources and Evaluation Conference (LREC 2010)*.


Abstract

Modern transformer-based language models are revolutionizing NLP. However, existing studies into language modelling with BERT have been mostly limited to English-language material and do not pay enough attention to the implicit knowledge of language, such as semantic roles, presupposition and negations, that can be acquired by the model during training. Thus, the aim of this study is to examine behavior of the model BERT in the task of masked language modelling and to provide linguistic interpretation to the unexpected effects and errors produced by the model. For this purpose, we used a new Russian-language dataset based on educational texts for learners of Russian and annotated with the help of the National Corpus of the Russian language. In terms of quality metrics (the proportion of words, semantically related to the target word), the multilingual BERT is recognized as the best model. Generally, each model has distinct strengths in relation to a certain linguistic phenomenon. These observations have meaningful implications for research into applied linguistics and pedagogy, contribute to dialogue system development, automatic exercise making, text generation and potentially could improve the quality of existing linguistic technologies.

1 Introduction

As is well-known, 2018 saw a breakthrough in natural language processing (NLP) with the advent of several novel pre-trained language models, including BERT (Devlin et al, 2018). These models are capable of fine-tuning, that is, additional training on a smaller dataset for a specific task. Such models develop a certain degree of understanding natural language and therefore require further studies (Zhang et al, 2019; Wallat et al, 2020). To this end, the so-called probing tasks are commonly used, such as testing the model’s ability to identify semantically coherent sentences in running text. A considerable amount of recent research (Gulordava et al, 2018; Salazar et al, 2019, Sun et al, 2019) was devoted to the general language modelling problem and also to masked language modelling. Other scholars (Devlin, 2018; Gong, 2019; Rogers et al, 2020) explored BERT specifically. However, there is a lack of studies concerning BERT’s behavior on Russian language material. In addition, existing papers discuss the problem without taking into consideration systematic linguistic features.

This paper aims to explore the behavior of the pre-trained BERT language model using the example of the diagnostic task of masked language modeling for the Russian language and give a linguistic interpretation of the cases when the model shows unsatisfactory results. This study will focus on new language data with experiment design based on linguistic theories1. The problem will be discussed in terms of the language modelling concept, cognitive science, theory of language, in particular the language frames theory, semantic roles concept, and also negations processing depending on a context.

2 Related Work

Bengio and co-authors (Bengio et al, 2003) suggest the following definition of a language

1 All datasets and the Jupyter notebook with the experiments are made available as a GitLab repository: https://gitlab.com/lieutkat/linguistic-experiments-with-bert/
model: probability distribution over word sequences. Such models can be the basis for solving a large number of NLP tasks, with the help of fine-tuning the general language models. The pre-trained language model weights already encode a lot of information about natural language. Specifically, a pre-trained model provides features of semantics, syntax, and “verbal knowledge” which may be transferred from general to specific tasks (Roberts et al, 2020; Manning et al, 2020). In the past several years, the BERT model and its modifications have become widely used in the natural language processing community (Liu et al, 2019; Lan et al, 2019). The model can be applied to all general types of tasks: single sentence prediction, sentence pair classification, question answering and sentence tagging. As BERT is inherently an encoder for language information, it often plays the role of a text feature extractor (Rogers et al, 2021).

Some authors (Conneau et al, 2018; Kim et al, 2019) focus on different types of tasks regarding language models, namely probing tasks. These authors suggest options for surface information (e.g., recovering a word from its embedding), syntactic information (e.g., the model sensitivity to word order change), semantic information (e.g., identification of main-clause verb tense). Their basic application is testing the pre-training effect: how pre-trained models encode various language phenomena. Currently, there are lists of linguistic capabilities available that allow to explore the of behavior of language models (Ribeiro et al, 2020). Thereby, the main benefit of probing tasks is the analysis of linguistic knowledge that can be extracted from sentence representations (Hewitt et al, 2019).

Other authors (Devlin et al, 2018; Song et al, 2019) suggested a new probing task for language models that is nowadays known as masked language modelling. The key difference is that the model masks some random words from a sequence and then predicts them again based on the contextual information, both left and right. Ultimately, it was suggested to train with masked language modeling, and then fine-tune for specific tasks (Wu et al, 2019; Salazar, 2019).

Differences in human and machine understanding of language were investigated in another paper (Ettinger, 2020), which inspired our experimental setup. The author uses psycholinguistic tests to find out the sensitivity of the model to such phenomena as hypernymy, semantic roles and negation. The tests revealed weaknesses in the language model and proved that computational models have great potential for natural language understanding.

As was pointed out in the previous section, despite this scholarly interest for BERT, there is a lack of studies with a strong linguistic basis. Additionally, previous studies have tended to focus primarily on English language material. The present study is based on a Russian-language dataset and uses theories from cognitive linguistics and the theory of language.

3 Linguistic Basis

From the linguistic perspective, the work of language models is based on the concept of semantic roles (Fillmore, 1976). Speaking about the subject of action, it can have both an agentive (or active) position, and a non-agentive (or inactive) position (Uskova, 2012). Ch. Fillmore also developed the theory of presupposition, which is important for this study. It is the preliminary knowledge that is responsible for the semantic correctness of the utterance. For example, the sentence Snow is expected in Moscow on February 30th, is incorrect due to the fact that it includes a false presupposition on February 30th.

Meanwhile V.Z. Demyankov identifies several types of presuppositions. In our experiment, we will use pragmatic and logical types. Pragmatic presupposition is the conditions and contexts that must be present in order for the speaker’s intention to be realized correctly. The semantic presupposition characterizes the relationship between a sentence and the proposition it expresses.

Another linguistic aspect that requires our consideration is negation. It is considered a semantic primitive integrated into the grammatical and lexical systems of all languages of the world (Paducheva, 2011). We will explore predicate negation.

Speaking about studies of language models, it is necessary to mention the type of lexical paradigmatic relations, such as hypo-hyperonymic. This type of relations reflects the direction of human thinking to systematize lexical units and non-linguistic structures behind them and bring them to a hierarchical form (Kuznetsova, 1989). This fact can be used to evaluate the performance of the language model in such a way that the resulting metric value is interpretable.

So, after considering some linguistic theories and concepts, we came to the term “behavior of
the language model”. According to the Philosophical Encyclopedia (Ilyichev et al, 1983), behavior is a way of reacting to any influence. In this paper, the behavior of a language model is understood as text data that the model provides as output under certain conditions of its use, namely, when testing or applying to a problem.

4 Datasets and Methods

The final dataset consists of three parts, each containing 50 sentences, which were collected using a linguistic observation method. Each part was designed to test and evaluate a certain aspect of the functioning of the language model, namely: common sense inference, the interpretation of semantic roles, and processing negatives depending on the context. Table 1 shows a sample of the data for semantic role interpretation.

<table>
<thead>
<tr>
<th>content</th>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>В процессе разговора я вдруг заметил, что она частенько [MASK] переводит на украинский язык. During our conversation, I suddenly noticed that she often [MASK] to Ukrainian.</td>
<td>переходит switched</td>
</tr>
<tr>
<td>В процессе разговора я вдруг заметил, что украинский язык частенько [MASK] в ее речи. During our conversation, I suddenly noticed that the Ukrainian language was often [MASK] by her.</td>
<td>слышится used</td>
</tr>
<tr>
<td>Она [MASK] местным украинским землячеством. She [MASK] the local Ukrainian community.</td>
<td>руководила leads</td>
</tr>
<tr>
<td>Местное землячество [MASK] ею. The local Ukrainian community was [MASK] by her.</td>
<td>руководилось led</td>
</tr>
</tbody>
</table>

Table 1. Example of a subcorpus for checking the interpretation of semantic roles.

There were two main resources for collecting data. Firstly, educational texts for teaching Russian as a foreign language for B1+ learners, with the help of which the “content” column was filled. Secondly, the National Corpus of the Russian language (RNC)², in which the expected words for contexts were selected through a semantic search. For instance, the selection of contexts for nouns with time semantics was carried out by entering the following characteristics: r:abstr & (t:time:period | t:time:moment).

In the case of semantic roles, complex or simple common sentences were selected from educational texts, the predicate was replaced with a token mask, and the linguistic unit, which had the role of an agent, took on the role of a patient (or experiencer).

To check common sense inference, we selected two consecutive sentences from the educational texts where one word in the second sentence was manually replaced with a mask. Then, using semantic search in the RNC, we selected a word of a related topic and a word of a more general topic with respect to the expected topic. Next, the subcorpus with negations was compiled as follows: in the RNC, using semantic search, we selected sentences with an adjective at the end, and masked the adjective. Finally, the sentence was copied and transformed into a negative one, and the adjective was replaced by its antonym.

The dataset was examined using a Jupyter notebook in the Python programming language with the help of the libraries tensorflow, pytorch and transformers. To perform a comparison, the BERT DrMatters ³ and BERT DeepPavlov ⁴ models for the Russian language, and the multilingual BERT ⁵ were selected.

- The BERT model (bert-base-multilingual-cased) includes 12 layers, 768 hidden layers, 12 heads of the attention mechanism and 179 million parameters. Trained on Wikipedia texts for 104 languages.
- The RuBERT (or BERT DeepPavlov) model was created by the team of the Moscow Institute of Physics and Technology. It contains 12 layers, 768 hidden layers, 12 heads of the attention mechanism and 180 million parameters. The model was trained on the Russian-language Wikipedia and news.
- BERT DrMatters model is based on the BERT DeepPavlov. There is a lack of information about all the other characteristics.

For the assessment, the RuWordNet model was loaded using the ruwordnet package. We received lists of hyperonyms and hyponyms for the target

---

² To access the RNC search interface see https://ruscorpora.ru/old/en/search-main.html
³ See https://huggingface.co/DrMatters/rubert_cased
⁴ See http://docs.deeppavlov.ai/en/master/features/models/bert.html
⁵ See https://huggingface.co/bert-base-multilingual-cased
word, then checked which of the prediction words are included in this list and counted them. Subsequently, the resulting list of numbers was normalized for convenient processing. The obtained data can be interpreted as the proportion of words in the prediction that are semantically related to the target word (that is, there are hyper-hyperonymic relations between them). Using such a metric, it is possible to understand whether the model is trying to put the correct group of objects in the place of the mask, that is, to test the model's ability to generalize and differentiate.

5 Results

5.1 Quality of Predictions

Table 2 provides information about the quality of the models’ predictions for several types of experiment, the measure is normalized RuWordNet-based. The largest values for each type of experiment are highlighted in gray.

<table>
<thead>
<tr>
<th>aspect</th>
<th>BERT Multilingual</th>
<th>BERT DeepPavlov &amp; DrMatters</th>
</tr>
</thead>
<tbody>
<tr>
<td>common sense</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>semantic roles</td>
<td>0.12</td>
<td>0.1</td>
</tr>
<tr>
<td>negations</td>
<td>0.12</td>
<td>0.1</td>
</tr>
<tr>
<td>negations (aff)</td>
<td>0.17</td>
<td>0.15</td>
</tr>
<tr>
<td>negations (neg)</td>
<td>0.17</td>
<td>0.15</td>
</tr>
</tbody>
</table>

Table 2. Measures based on RuWordNet for multilingual and Russian models (aff – affirmative context, neg – negative context).

The advantage of the measure based on RuWordNet is the fact that a successful prediction is not only the target word itself, but also many of its hyponyms and hyperonyms. For this reason, the values tend to be quite interpretable. Overall, the multilingual model in experiments with semantic roles and negations is slightly ahead of both Russian ones, which have equal values for each aspect. Moreover, no differences were found between the processing of the two types of contexts in the negation experiment. The results of the linguistic analysis are presented below.

5.2 Linguistic Analysis: Common Sense Inference

In case of multilingual BERT, the influence of the pragmatic presupposition is strong, while the role of the logical presupposition is minimal (see Table 3, example 1). It can be assumed that the model uses the acquired background knowledge about historical facts. At the same time the demonstration of background knowledge dominates the observance of grammatical correctness of the prediction (see Table 3, example 2).

<table>
<thead>
<tr>
<th>sentence</th>
<th>target</th>
<th>model's predictions</th>
</tr>
</thead>
</table>

Table 3. Examples from qualitative analysis, multilingual BERT (common sense).

Both models for the Russian language have problems with the pragmatic presupposition: sometimes it is insufficient to fulfill the predictions. Inaccuracies in the predictions occur under the influence of the logical presupposition when it is more pronounced than the pragmatic one (see Table 4). In the example below, the models incorrectly interpret the logical presupposition and perform a pronominal replacement of the subject (“the boy”).

<table>
<thead>
<tr>
<th>sentence</th>
<th>target</th>
<th>model's predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Рост Наполеона был выше среднеевропейского. Историки давно закрепили за французским [MASK] прозвище Маленький Капрай.</td>
<td>полководец</td>
<td>[&quot;Наполеон&quot;, &quot;человек&quot;, &quot;племя&quot;, &quot;орден&quot;, &quot;так&quot;, &quot;имя&quot;, &quot;он&quot;, &quot;этот&quot;]</td>
</tr>
</tbody>
</table>

Table 4. Examples from qualitative analysis, multilingual and Russian models.
The boy took out a case, pulled out a gun and began to hurriedly choosing the right ones.

Table 4. Examples from qualitative analysis, BERT DeepPavlov and BERT DrMatters (common sense).

5.3 Linguistic Analysis: Interpretation of Semantic Roles

In the predictions of the multilingual BERT model, the target words themselves are almost absent, but there are more common synonyms for them (see Table 5, example 1). For the case of the non-agentive position of the subject, the model is practically unable to predict the target word correctly (see Table 2, example 2).

Table 5. Examples from qualitative analysis, multilingual BERT (semantic roles).

As for BERT DeepPavlov and BERT DrMatters, when the subject is in the agentive position, the predictions often include the target word (see Table 6, example 1). Moreover, in the situation of the non-agentive position of the subject, the short length of the sentence negatively affects the predictions (see Table 6, example 2).

Table 6. Examples from qualitative analysis, BERT DeepPavlov and BERT DrMatters (semantic roles).

5.4 Linguistic Analysis: Negations Depending on the Context Type

Multilingual BERT’s predictions contain a lot of noise in the form of subwords, UNK-tokens, and punctuation marks. In the predictions without noise, there are often explicit or implicit negation components (e.g., “no”, “absent”, “lack”). Thus, we can suppose that at a high level of abstraction, the model has acquired information about the nature of the context, but it does not have enough representations to express it (see Table 7).
7 Conclusion and Future Work

This paper has investigated the pre-trained BERT language models with probing tasks. As far as we are aware, this is the first time that BERT was explored by professional linguists based on Russian-language material. As a result of this research, we have listed the frequent mistakes made by the BERT model in the masked language modelling task and conducted their analysis. In contrast to previous studies (Ettinger, 2020), where the focus was on psycholinguistic diagnostics and comparison between machine and human performance, we made our study more linguistically grounded by adding knowledge from theory of language. While conducting the research, we created a unique dataset that represents several cognitive phenomena with special annotations. The dataset is made available to the community. In the future, we plan to expand the dataset and annotations to cover more cognitive and linguistic aspects.

Acknowledgments

This work is supported by RSF (Russian Science Foundation) grant 20-71-10010.

References


Towards Quantifying Magnitude of Political Bias in News Articles Using a Novel Annotation Schema

Lalitha Kameswari, Radhika Mamidi
Language Technologies Research Centre
IIIT Hyderabad
Telangana, India
v.a.lalitha@research.iiit.ac.in
radhika.mamidi@iiit.ac.in

Abstract

Media bias is a predominant phenomenon present in most forms of print and electronic media such as news articles, blogs, tweets, etc. Since media plays a pivotal role in shaping public opinion towards political happenings, both political parties and media houses often use such sources as outlets to propagate their own prejudices to the public. There has been some research on detecting political bias in news articles. However, none of it attempts to analyse the nature of bias or quantify the magnitude of the bias in a given text.

This paper presents a political bias annotated corpus viz. PoBiCo-21, which is annotated using a schema specifically designed with 10 labels to capture various techniques used to create political bias in news. We create a ranking of these techniques based on their contribution to bias. After validating the ranking, we propose methods to use it to quantify the magnitude of bias in political news articles.

1 Introduction

An increase in Internet and social media access has made it easier for people to know the happenings from all parts of the world. This improved reach of the media has led to people and organisations utilising such platforms to increase their popularity. With this extensive usage of mass communication, there is an increased risk of misinformation, fake news, twisting of facts, and the spread of controversies. In such cases, it is the responsibility of the media houses to verify the authenticity of the reported information and ensure that it is complete, neutral, and not misleading.

However, we often see instances where some news articles emphasize more on particular viewpoints selectively. In some cases, journalists and media houses either intentionally or subconsciously present biased information aligned with their own political ideologies. This can alter public opinion hugely and impact the decision of the general public. This phenomenon of political bias is very prevalent in democracies, especially during elections, where influencing public opinion can have far-reaching consequences.

Detection of political bias in the news is a complex task when it comes to a multi-party political system like India, where multiple parties operate at both state and national levels\(^1\). There has been limited work to date in this area, as discussed in Section 2. However, in a scenario where most of the political content contains bias in some form, it is not just sufficient to identify the existence of bias.

Therefore, we propose two important contributions through this paper - a fine-grained annotation schema specifically designed for analysing political bias and a new sentiment analysis based approach to quantify the magnitude of political bias in an article. We also present PoBiCo-21, an annotated corpus for political bias, containing 500 news articles along with their headlines annotated for various aspects of political bias using our novel annotation schema as discussed in Section 3. The dataset contains news articles in Telugu\(^2\), which is a low resource Dravidian language spoken primarily in the states of Telangana and Andhra Pradesh in India.

2 Related Work

The area of political discourse analysis is closely associated with media studies, political science, anthropology, sentiment analysis, and opinion mining. However, most of the work done is limited to English, primarily due to the high availability of data. For example, Jiang and Argamon (2008) identify

\(^1\)https://bit.ly/3p74aHx
\(^2\)https://en.wikipedia.org/wiki/Telugu_language
subjective sentences in blog posts to understand the political orientation of a given text. Nasab and Dowlatabadi (2015) proposed a rule-based linguistic method to identify bias in a news article based on the semantics of the headline and the article.

Gangula et al. (2019) and Cruz et al. (2020) used attention mechanism on different kinds of document representations to identify bias in an article. Kameswari et al. (2020) proposed a hybrid approach to improve the performance of such ML models by establishing a correlation between presupposition and bias.

Fan et al. (2019) pointed out that restricting to lexical level for detecting bias might not be sufficient and put forward a method of identifying spans of bias by looking at sentences/clauses which are speculative or tangential to the main point being talked about.

Zhou et al. (2011) made use of the comments and opinions of readers in order to classify articles as liberal or conservative. But the limitation to this approach is that it cannot perform well in a political setting where multiple alliances and several parties operate at different levels of the government.

All the work discussed above deals only with the detection of bias, and none of these approaches give us any further insight into the nature or the magnitude of bias in a given text. Our work fills this gap by creating a specialised fine-grained annotation schema for analysing and quantifying the political bias in a given text.

3 Data and Annotation

Our first requirement was a dataset of political news articles containing positive and negative bias along with some unbiased articles. For this purpose, we used the public dataset created by Gangula et al. (2019) as our primary data to begin with. Our aim is to annotate the dataset with different techniques through which bias is generated, and identify the levels of reporting at which it happens. This will be discussed in detail in Section 3.1. We removed the articles which were very short, or had no clear target of bias, either positive or negative. The statistics of our dataset are presented in Table 1.

3.1 Political Bias Techniques

Bias can be incorporated in news articles in several ways. It ranges from selection or gatekeeping bias, which is the selection and filtering of news broadcasted by media houses, since the set of world events is too large to be treated exhaustively. In the case of politics, the scope for subjectivity in selecting information about some specific event, person, or political party often induces bias due to resource constraints, editorial guidelines, ideological affinities, or even the fragmented nature of the information at a journalist’s disposal (Bourgeois et al., 2019).

In this paper, we focus primarily on Presentation Bias which shows how the way of presenting some information can influence the readers directly or indirectly. Since the objective of introducing bias is to persuade the reader towards or against an entity, the methods used to create such bias can also be considered as propaganda. Political Scientist Harold Laswell defines propaganda as “the expression of opinions or actions carried out deliberately to influence the opinions or actions of individuals for predetermined ends and through psychological manipulations”. Herman and Chomsky (2010) came up with a theory on propaganda which views mass media as a group of businesses whose job is to sell readers and audience as a product to other advertising businesses primarily controlled by the government and the corporate sectors.

Based on the strong correlation between propaganda and bias, we used the list of all propaganda

\begin{table}[h]
\centering
\begin{tabular}{|l|c|}
\hline
Polarity of Bias & Number of Articles \\
\hline
Positive & 180 \\
Negative & 220 \\
Unbiased & 100 \\
\hline
Total & 500 \\
\hline
\end{tabular}
\caption{Statistics of our dataset}
\end{table}

3https://en.wikipedia.org/wiki/Sakshi_Telugu

Telugu newspaper owned by the family of the Y.S. Jagan, the leader of YSRCP party and current Chief Minister of Andhra Pradesh. Other possible reasons include the incorporation of subjectivity, or the influence of the personal ideologies of the journalists, editors, and writers of an article, despite the ideal expectation of objective reporting.
techniques compiled from Propaganda and Mass Persuasion: A Historical Encyclopedia, 1500 to the Present by Grandstaff (2006). The list is available on Wikipedia. The original list contains 68 techniques of propaganda. Out of these, many of them were not applicable to political bias. We removed such techniques and clubbed some very similar techniques in the context of political bias to finally create a list of 10 techniques discussed below. It is important to note that these techniques are not mutually exclusive - a given sentence/article may contain multiple techniques together.

1. **Cultural/Identity Bias**: This kind of bias includes all sorts of prejudices towards the class and the identity of a politician/party based on several divisions. Some examples are a subtle or direct portrayal of discriminatory attitudes like racism, casteism, ageism, sexism, etc., which could either be used positively or negatively to create bias.

2. **Amplification**: This is a term used to refer to the introduction of bias by overly hyping or exaggerating information about events/people/parties in order to present a skewed impression to the readers. It is often identified by the use of loaded language in political news. It can cause misrepresentation of the truth of an event or alter the reader’s image of a politician/party/political event.

3. **Personal Targeting**: This technique is said be used when an individual is attacked/glorified for his personal traits instead of their political contributions. This is often done as an attempt to divert attention from other issues, or sometimes even as a way to show agreement/disagreement towards some policies or situations.

   Ad Hominem or Personal attack is a technique of attacking some individuals or their political parties instead of attacking their arguments and policies. This can include name calling, stereotyping, labelling and scapegoating. The opposite of Personal attack is Personal praise, where a person’s qualities and virtues are glorified instead of their work/policy/contribution. It is very common to see politicians engaging in personal attacks towards the leaders of other parties, and highly praising their own party leaders.

4. **Repetition**: Ad Nauseam or Repetition is a way of introducing bias by repeating an idea, question, or a slogan in an attempt to sub-consciously establish that as the truth for the reader.

5. **Appeal to Audience**: This is a technique where the audience is persuaded or won over by appealing to various aspects of their lives such as their fears, beliefs and interests. This technique most commonly occurs in the following ways:
   - **Appeal to fear**: This happens when media or a politician seeks to build support of the audience by instilling panic and fear in them, usually about the outcome of not following the suggested course of action.
   - **Appeal to beliefs and prejudices**: This happens when media or a politician seeks to build a connection with the audience by bringing up shared beliefs and prejudices, and using them to influence the opinion of the audience. This often happens when politicians want to alienate or create a distance between the audience and any other party, by portraying it as an “outside entity” which does not fit in their shared beliefs and opinions.
   - **Appeal to shared identity**: This happens when media or a politician appears to be a “part” of some aspect of the identity of the intended audience, and highlights the image of them as a part of that shared identity.
     Unlike beliefs, these aspects of identities are more concrete. For example, a politician dressing up like a farmer and sowing some crops while campaigning and appealing to farmers.
   - **Appeal by making promises**: This includes trying to persuade a specific set of audience by making promises which benefit them.

6. **Bandwagon**: This is a technique where the audience is persuaded to follow a particular path by presenting it as the most favourable
path which all others are taking. This is done by assuring good results, such as guaranteed victory in the elections.

7. **Black and White Fallacy/Rhetorical Questions**: Black and White fallacy involves presenting the audience with only two choices, while clearly showing only one of them as the most suitable choice. In case of political news, this is also seen in the form of rhetoric yes/no questions, where only one of the answers seems logically correct in that particular context.

Another similar way of using questions to persuade someone is called Hypophora, where the audience is presented with a question, and the speakers themselves give a seemingly correct answer along with an explanation, without giving the audience much time to contemplate. This is done to subconsciously make the audience accept the proposed answer as the correct one without questioning it further.

8. **Intentional Vagueness**: Intentionally presenting the audience with vague information, unstated assumptions or unsupported claims falls under this category. This is often done to make the audience draw certain expected conclusions, which would have been difficult if they knew the whole scenario clearly.

For example, there has been study conducted by Kameswari et al. (2020) which shows how introducing vagueness in the form of presupposition contributes to bias in an article.

9. **Oversimplification**: The idea behind this technique is to oversimplify the implications of following/not following a course of action and showing that it leads to some non-acceptable outcome.

This is similar to a philosophical argument called Reductio ad absurdum, where one attempts to establish a claim by showing that the opposite scenario would lead to absurdity or contradiction.

These ways are often used to persuade the audience to support a presented idea, person or a political party without directly asking them to do so.

10. **Whataboutism**: This is a logical fallacy used to discredit an opponent’s claim by deflecting it to something else. This mostly happens in the scenarios where one is questioned about his/her own actions. There have been studies by Islam (2018) and Dykstra et al. (2020) which show how politicians use whataboutism to dodge questions.

For instance, when the former US President Donald Trump was asked about his opinion on whether Virginia should keep a statue of Confederate general Robert E. Lee, he did not give a straightforward answer but resorted to this technique by asking, “What about other statues of famous Americans?”

### 3.2 Annotation Guidelines and Procedure

Each article was already annotated for the polarity of the bias (-1 for negative, 1 for positive and 0 for unbiased) in the corpus we created from the corpus of Gangula et al. (2019). We additionally had the information of the newspaper each article was taken from. There were 3 newspapers from which the articles were taken - Eenadu, Andhrrajyothy and Sakshi.

![Figure 1: Guidelines for Fine Grained Annotation](https://time.com/4941771/donald-trump-whataboutism-rhetoric/)

Each article was annotated for the presence of the 10 techniques described in Section 3.1. Instead of binary annotation (0 or 1 for the presence or absence of a technique), we decided to annotate at a more fine grained level. This was done to capture whether a particular technique was directly quoted, or introduced indirectly while reporting. This kind of indirect incorporation of bias by media while reporting is known as **Spin**, which was found to
be a frequently occurring phenomena employed to twist the facts and influence public opinion (Mullainathan and Shleifer (2002), Burke (2008)).

For each technique, the annotator had to label 0 if it was not present anywhere in the article, 1 if it was quoted directly in any part of the article, 2 if it was indirectly incorporated in the article, and 3 if it was indirectly incorporated and present in the headline of the article. These guidelines are shown in a hierarchical decision tree format for clear understanding in Figure 1. Each article was annotated by 2 annotators, and the inter-annotator agreement was found to be 0.715.

3.3 Statistics

The statistics of the dataset are presented in Table 2. We observe that the direct occurrences are higher than indirect and headline occurrences for most of the techniques. Personal targeting is the most frequently used technique followed by amplification and intentional vagueness. Oversimplification and repetition are the least frequent techniques in the dataset.

4 Sentiment Based Ranking

Our first step towards quantifying the magnitude or severity of the political bias in an article is to create a ranking of the techniques mentioned in 3.1 based on how much they contribute to creation of bias in a text. This was inspired by Graham’s Hierarchy of Disagreement which classifies disagreement, particularly on online platforms, into 7 levels as shown in Figure 2.

The least level of disagreement was Name calling, where a person just says repulsive things about another person or group of people. On the other end, the highest level of disagreement was Refuting the central point of an argument.

We claim that the sentiment of an article can be a useful indicator of the severity of political bias, since an unbiased article contains only objective information devoid of positive or negative sentiment. Based on this claim, we calculate the sentiment of each news article as the average of its sentence-level sentiment values (-1, 0, 1).

4.1 Model

For the purpose of sentiment analysis in Telugu, we use IndicBERT (Kakwani et al., Kunchukuttan et al.) which is a multilingual NLU model pretrained on 12 Indian languages and evaluated on IndicGLUE. The model used is ALBERT (Lan et al., 2019), a light-weight, compact version of BERT with fewer parameters. After generating embeddings for each sentence using the model, we follow the same approach as Kunchukuttan et al. (2020) and use a KNN classifier with k=4 for sentiment classification task. This performed with an accuracy of 0.52 and F1 score of 0.54 on ACTSA News dataset (Mukku and Mamidi, 2017), which

---

Table 2: Statistics of the number of occurrences of each technique in the dataset

<table>
<thead>
<tr>
<th>Technique</th>
<th>Occurrences in negatively biased articles</th>
<th>Occurrences in positively biased articles</th>
<th>Occurrences in neutral articles</th>
<th>Direct occurrences (Label=1)</th>
<th>Indirect occurrences (Label=2)</th>
<th>Headline occurrences (Label=3)</th>
<th>Total occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cultural/Identity Bias</td>
<td>110</td>
<td>40</td>
<td>8</td>
<td>140</td>
<td>10</td>
<td>8</td>
<td>158</td>
</tr>
<tr>
<td>Amplification</td>
<td>285</td>
<td>75</td>
<td>5</td>
<td>245</td>
<td>50</td>
<td>69</td>
<td>365</td>
</tr>
<tr>
<td>Personal Targeting</td>
<td>325</td>
<td>95</td>
<td>2</td>
<td>305</td>
<td>20</td>
<td>99</td>
<td>422</td>
</tr>
<tr>
<td>Repetition</td>
<td>75</td>
<td>15</td>
<td>4</td>
<td>35</td>
<td>55</td>
<td>4</td>
<td>94</td>
</tr>
<tr>
<td>Appeal to audience</td>
<td>165</td>
<td>55</td>
<td>17</td>
<td>210</td>
<td>13</td>
<td>14</td>
<td>237</td>
</tr>
<tr>
<td>Bandwagon</td>
<td>125</td>
<td>35</td>
<td>4</td>
<td>150</td>
<td>10</td>
<td>4</td>
<td>164</td>
</tr>
<tr>
<td>Black and White Fallacy/Rhetorical Questions</td>
<td>145</td>
<td>25</td>
<td>19</td>
<td>150</td>
<td>15</td>
<td>24</td>
<td>189</td>
</tr>
<tr>
<td>Intentional Vagueness</td>
<td>275</td>
<td>75</td>
<td>8</td>
<td>205</td>
<td>50</td>
<td>99</td>
<td>358</td>
</tr>
<tr>
<td>Oversimplification</td>
<td>55</td>
<td>11</td>
<td>5</td>
<td>45</td>
<td>12</td>
<td>14</td>
<td>71</td>
</tr>
<tr>
<td>Whataboutism</td>
<td>120</td>
<td>15</td>
<td>11</td>
<td>120</td>
<td>15</td>
<td>11</td>
<td>146</td>
</tr>
</tbody>
</table>
is comparable to current state-of-the-art text classification performance in Telugu.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Avg Article Sentiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal Targeting</td>
<td>0.169</td>
</tr>
<tr>
<td>Amplification</td>
<td>0.135</td>
</tr>
<tr>
<td>Intentional Vagueness</td>
<td>0.122</td>
</tr>
<tr>
<td>Bandwagon</td>
<td>0.113</td>
</tr>
<tr>
<td>Black and White Fallacy/Rhetorical Questions</td>
<td>0.093</td>
</tr>
<tr>
<td>Appeal to Audience</td>
<td>0.090</td>
</tr>
<tr>
<td>Whataboutism</td>
<td>0.078</td>
</tr>
<tr>
<td>Cultural/Identity Bias</td>
<td>0.073</td>
</tr>
<tr>
<td>Repetition</td>
<td>0.059</td>
</tr>
<tr>
<td>Oversimplification</td>
<td>0.042</td>
</tr>
</tbody>
</table>

Table 3: Average values of the sentiment scores of articles containing each technique in descending order.

4.2 Results

Using the sentiment values for each article in the annotated dataset, we try to compute the average sentiment value corresponding to each political bias technique. Since our primary interest is to see how much a technique can shift the article from neutrality, we consider only the magnitudes of the sentiment values of each articles. We collect all the articles which contain a particular technique, and calculate the sentiment score of the technique as the average of all the absolute values of the article sentiments.

According to initial claim, the higher the sentiment value of a technique, the higher is its contribution towards creating political bias in an article. The sentiment value for each technique is shown in Table 3. Personal Targeting has the highest value, which can be supported by the fact that attacking or praising an individual/party instead of focusing on policies/actions is the most direct way of introducing bias. It can also be observed from Table 2 that Personal Targeting was observed in around 84% of the articles in our dataset, making it the most common and frequently used technique in Political bias. Figure 3 shows the sentiment ranking of the techniques in a graph.

5 Schema Validation

To test the reliability of the schema proposed in Section 4.2, we tested it on a bias magnitude comparison task against the human annotation results. For this purpose, we created a test dataset consisting of 100 article pairs. For each pair, the task was to identify the more biased article of the two. All the articles were taken from the political news domain. For human annotation, we had the annotation done by two annotators who had no idea about the schema. We removed the articles on which there was a disagreement between the two annotators. We were left with 92 articles.

Then we gave the test pairs to two other annotators who had the list of the 10 techniques along the definition, explanation and examples. Their task was to identify all the techniques present in the two articles given in each article pair. Then they were instructed to give bias scores for each article by adding the scores of all the techniques from Table 3. In a given pair, the article having a higher score was labelled as the one having a higher magnitude of bias.

If both the articles in a pair had same bias score but at least one non-common label, the comparison was to be done based on the non-common label. If two articles had the same score and same labels, we asked the annotators to mark the shorter of the two articles as the more biased one.

We found that 66 out of 92 times, we were able to correctly identify the more biased article using our ranking. This gives us confidence that the ordering of techniques done through our ranking has a strong correlation with the relative contribution of each technique towards political bias in an article.

6 Conclusion and Future Work

This paper explores the creation of a novel annotation schema which captures the nuances of political bias in a fine grained manner. We propose a sentiment analysis based ranking of 10 political bias techniques, and a validation study to show that this ranking corresponds to the relative contribution of each of these techniques towards political bias. We also contribute a fully annotated dataset- PoBiCo-21, containing 500 articles annotated using this schema. The fine grained annotation can also be used for making other interesting observations from political news data such as identifying media houses which often use biased/ misleading headlines, mining frequently co-occurring techniques of political bias, etc.
6.1 Proposed Direction for Future Research

In the future, we can create an end-to-end bias scoring system which takes an article as an input, tokenises it, detects all potential spans of text containing political bias, assigns the relevant political bias techniques to each span, and computes the final score as the summation of each span length multiplied by the relative sentiment scores of the techniques present in that span. This is shown in a step-by-step manner with an example in Figure 4.

Detection of bias spans can be treated as a standard sequence labelling problem. With good amount of labelled data, we can use Neural Transformer architectures like BERT (Devlin et al., 2018), or its variants like RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019) etc. Once that is done, we can do a one-to-many mapping from spans to the political bias techniques and then compute the final score of the article.

With the help of this system, a reader can not only quantify and compare the magnitude of political bias of several articles, but also identify the techniques which are being used either directly (by the politicians) or indirectly (by the media) to create bias.

References


André Ferreira Cruz, Gil Rocha, and Henrique Lopes Cardoso. 2020. On document representations for detection of biased news articles. In Proceedings of...


Application of Mix-Up Method in Document Classification Task Using BERT

Naoki Kikuta and Hiroyuki Shinnou
Faculty of Engineering, Ibaraki University, Japan
{21nm720n, hiroyuki.shinnou.0828}@vc.ibaraki.ac.jp

Abstract
The mix-up method (Zhang et al., 2017), one of the methods for data augmentation, is known to be easy to implement and highly effective. Although the mix-up method is intended for image identification, it can also be applied to natural language processing. In this paper, we attempt to apply the mix-up method to a document classification task using bidirectional encoder representations from transformers (BERT) (Devlin et al., 2018). Since BERT allows for two-sentence input, we concatenated word sequences from two documents with different labels and used the multi-class output as the supervised data with a one-hot vector. In an experiment using the livedoor news corpus, which is Japanese, we compared the accuracy of document classification using two methods for selecting documents to be concatenated with that of ordinary document classification. As a result, we found that the proposed method is better than the normal classification when the documents with labels shortages are mixed preferentially. This indicates that how to choose documents for mix-up has a significant impact on the results.

1 Introduction
The high cost of constructing training data is always a problem when solving natural language processing (NLP) tasks using machine learning approaches. Several attempts have been made to solve this problem. One of the most recent methods for constructing training data is data augmentation (Shorten and Khoshgoftaar, 2019). Data augmentation methods can be divided into two types: processing and generation. For image identification, even if an image in the training data is flipped or cropped, the image label does not change. This means that the training data can be increased by adding such processed images to the training data. Alternatively, the method for generating artificial data using generative adversarial network can be considered as a type of data augmentation. The mix-up method is one of the methods for generating data augmentation. It is highly effective and can easily be implemented. Although the mix-up method is used for image identification, it can also be used for NLP.

2 Related topic
2.1 Bidirectional encoder representations from transformers (BERT)
BERT is a high-performance, pre-trained model that has been widely used since its creation by Google (2018) (Devlin et al., 2018). It can be used for classification, word prediction and context determination. In this study, we improve the accuracy of the BERT-based document classification task using the mix-up method.

2.2 Mix-up method
Mix-up is a data augmentation method in the field of image proposed by Hongyi Zhang (2017) (Zhang et al., 2017). The data augmentation is performed using Equations 1 and 2 for image data and labels respectively.

\[ x = \lambda x_i + (1 - \lambda) x_j \]  
\[ y = \lambda y_i + (1 - \lambda) y_j \]

x is a vector of image data , y is a one-hot vector of labels, and \( \lambda \) is the mixing ratio.

3 Previous studies using mix-up method for NLP
Hongyu Guo (2019) conducted a study using the mix-up method for NLP (Guo et al., 2019). The data augmentation is performed using Equations 1 and 2 for image data and labels respectively.
4 Proposed method

If we adopt the mix-up method of the previous study for BERT, we will have a problem. In the methods of the previous study (Guo et al., 2019), it is necessary to create feature vectors of the documents before learning the neural network (NN) (Figure 2). However, document classification using BERT obtains the feature vectors of the documents during the NN learning process (Figure 1), which is a different order from the methods used in the previous study. If we were to adopt the method of the previous study, the calculation of feature vectors by BERT would be done outside the learning process. Therefore, high classification accuracy cannot be expected.

In this paper, we propose the following method.

4.1 How to mix data

We don’t use Equation 1 to mix data. Our method is to concatenate the word sequences of the two documents when they are entered into BERT. With this method, it is possible to learn the BERT part to obtain the feature vector of the document (Figure 3). We present the following examples. This time, we used Japanese document as target and Japanese BERT as model. Compared to English, there is no clear separation between words in Japanese. Therefore, when processing Japanese, it is necessary to divide it by tokenizer into words, characters, and other parts by toke. Then, each devided word is assigned an ID. The ID 2 indicates the beginning of the sentence and is not necessary for the following sequence, so it is excluded. Additionally, since the maximum input length for BERT is 512, we limited the first and second halves of the word sequences’ length to 252 each to avoid exceeding this value. If the sentence length exceeded 252, we discarded the remainder.

The first word sequence

[2, 6259, 9, 12396, 14, 3596, 3]

The second word sequence

[2, 11475, 9, 3741, 5, 12098, 75, 3]

Mixed word sentences

[2, 6259, 9, 12396, 14, 3596, 3,
11475, 9, 3741, 5, 12098, 75, 3]

4.2 How to mix labels
First, each label was represented by a one-hot vector consisting of 0 and 1. A vector consisting two 0.5 and seven 0 was created. The mixed labels contain 0.5; it indicates that the two documents are mixed equivalently. We have the following examples.

Label 3  
[0, 0, 0, 1, 0, 0, 0, 0, 0]

Label 6  
[0, 0, 0, 0, 0, 1, 0, 0, 0]

Mixing of labels 3 and 6  
[0, 0, 0, 0.5, 0, 0, 0.5, 0, 0]

5 Experiment

5.1 Conditions

5.1.1 Execution environment
The experiment was conducted using the graphics processing unit environment of Google Colaboratory.

5.1.2 BERT model we used
We used bert-base-japanese-whole-word-masking\(^1\), one of the pre-training BERT models for Japanese. It was developed by Inui and Suzuki Lab of Tohoku university.\(^1\)

\[\text{https://github.com/cl-tohoku/bert-japanese}\]

5.2 Experimental procedure

5.2.1 Preparing data
In this experiment, we extracted 6623 articles (texts) from the livedoor news corpus and sorted them as shown in Table 1.

\[\begin{array}{|c|c|c|c|c|}
\hline
\text{label} & \text{train} & \text{val} & \text{test} & \text{sum} \\
\hline
0 & 87 & 128 & 566 & 781 \\
1 & 87 & 125 & 571 & 783 \\
2 & 86 & 111 & 581 & 778 \\
3 & 51 & 72 & 335 & 458 \\
4 & 87 & 114 & 582 & 783 \\
5 & 84 & 106 & 565 & 755 \\
6 & 87 & 106 & 590 & 783 \\
7 & 90 & 131 & 589 & 810 \\
8 & 77 & 107 & 508 & 692 \\
\hline
\text{sum} & 736 & 1000 & 4887 & 6623 \\
\hline
\end{array}\]

Table 1: Breakdown of data used

5.2.2 Mix-up of training data
For the training data, we used the mix-up method to expand the data. We used the following two methods for selecting the documents to be mixed.

Selection method 1
The first method is to mix the documents of all labels randomly. We randomly sorted 736 documents

\[\text{https://www.rondhuit.com/download.html#ldcc}\]

\(^{1}\text{https://github.com/cl-tohoku/bert-japanese}\]

---

5.1.3 Corpus used
We used the livedoor news corpus\(^2\), which is Japanese to classify documents into the following nine labels.

- label 0: dokujo-tsushin
- label 1: IT lifehack
- label 2: Home Appliances Channel
- label 3: livedoor HOMME
- label 4: MOVIE ENTER
- label 5: Peachy
- label 6: smax
- label 7: Sports Watch
- label 8: topic news

\(^{2}\text{https://www.rondhuit.com/download.html#ldcc}\)
using a random number, and mixed two adjacent documents (and their labels) in order. We generated 735 extended data using this method. As a result, the number of training data was expanded from 736 to 1471. Additionally, each time the program is run, the selected combination changes.

Selection method 2
The second method of selection is to make up for documents with label shortages preferentially. In this experiment (Table 1), the training data lacks documents with label 3. Thus, we select label 3 documents to mix. Specifically, we randomly selected one document from 51 label 3 documents. Then, we randomly selected one document from 685 non-label 3 documents, and repeated the procedure of mixing the two documents. For the order of concatenation, the documents with and without label 3 form the first and second halves, respectively. As a result, the number of training data was expanded from 736 to 1501. Similar to the selection method (1), the combination changes every time the program is run.

5.2.3 The classifier we created
The model of the NN used as the classifier is BERT with an additional nn.Linear layer. We input a sequence of words of length 512 or less into BERT and obtain a feature vector of 768 dimensional documents from final layer as output. Then, we input it to nn.Linear layer Pytorch has and obtain the prediction for each label in nine dimensions as output. The detailed settings are shown below.

Loss function
Cross entropy: When training a classification problem with classes, nn.CrossEntropyLoss is usually used in Pytorch. However, this time, the labels are in one-hot representation and cannot be input directly into nn.CrossEntropyLoss. Therefore, we used LogSoftmax Pytorch has to calculate the loss according to the definition of cross-entropy (Equation 3). Since we used batch in this experiment, the loss is Equation 4 which is the batch average of Equation 3.

\[ E = - \sum_{k} t_k \log y_k \]  

(3)

\[ E = - \frac{1}{B} \sum_{b} \sum_{k} t_k \log y_k \]  

(4)

Here, \( t_k \) is a correct answer, \( y_k \) is a predicted value, and \( B \) is a batch size.

Optimization function
Stochastic gradient descent (SGD): Using the validation data, we set the learning rate to 0.01, considering both classification accuracy and learning efficiency (Figure 4).

Batch size of the training data
The batch size was set to 10, which was the maximum value possible in Google Colaboratory, the execution environment.

Number of epochs
Considering the range of increase in classification accuracy in the validation data (Figure 4), it was determined that the accuracy reached a convergence value after ten epochs of training.

![Figure 4: Percentage of correct answers in the validation data](image-url)
6 Result

For the normal BERT that doesn’t use mixup, the BERT with mix-up of selection method 1, and the BERT with mix-up of selection method 2, we prepared ten models trained with ten epochs of the training data for each of the methods. Then, the accuracy rate for the test data was calculated. Figure 5 show the box plots comparing each model. The comparison of the mean values of accuracy rate is presented in Table 2.

![Figure 5: Result](image)

<table>
<thead>
<tr>
<th></th>
<th>Accuracy rate (mean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>0.889</td>
</tr>
<tr>
<td>Mix-up selection 1</td>
<td>0.887</td>
</tr>
<tr>
<td>Mix-up selection 2</td>
<td>0.893</td>
</tr>
</tbody>
</table>

Table 2: Comparison of mean

In the order of increasing accuracy, there are mix-up for selection method 2, normal BERT, and mix-up of selection method 1.

7 Consideration

Given the result, we obtained that the selection method of the documents to be mixed has a great influence on the accuracy. In this experiment, it is effective to prioritize mixing documents with labels shortages. We would try different methods and conclude. In this experiment, we used two documents with equivalent values (ratio 0.5 : 0.5). However, we think that it is worthwhile to try a method for varying the length of the concatenated words. Mix-up method is easier to implement than other data augmentation methods in NLP, and its accuracy has been improved. It is expected to become a mainstream method in the future.

8 Conclusion

In this paper, we applied the mix-up method to a document classification task using BERT. Since BERT allows for two-sentence input, two documents with different labels were combined and input. Then, the labels were mixed by creating two 0.5 elements in a one-hot vector. In an experiment using the livedoor news corpus, which is Japanese, we found that the proposed method is better than the normal classification when the documents with labels shortages are mixed preferentially. Therefore, it indicates that the accuracy varies depending on the method of selecting documents to be mixed.

Acknowledgment

The research was supported by JSPS KAKENHI Grant Number JP19K12093 and ROIS NII Open Collaborative Research 21FC05.

References


Translation Memory Retrieval Using Lucene

Kwang-hyok Kim¹, Myong-ho Cho¹, Chol-ho Ryang¹, Ju-song Im¹, Song-yong Cho¹, Yong-jun Han²

¹Faculty of Information Science Kim Il Sung University Pyongyang, DPRK
kh.kim0107.ch.ryang0415.sy.jo1020@ryongnamsan.edu.kp
²Faculty of Foregin Language and Literature, Kim Il Sung University Pyongyang, DPRK
flit1@ryongnamsan.edu.kp

Abstract
Translation Memory (TM) system, a major component of computer-assisted translation (CAT), is widely used to improve human translators’ productivity by making effective use of previously translated resource. We propose a method to achieve high-speed retrieval from a large translation memory by means of similarity evaluation based on vector model, and present the experimental result. Through our experiment using Lucene, an open source information retrieval search engine, we conclude that it is possible to achieve real-time retrieval speed of about tens of microseconds even for a large translation memory with 5 million segment pairs.

1 Introduction
Translation memory technique is a key functionality being widely used in the field of CAT. Translation Memories (TMs) are “structured archives of past translations” which store pairs of corresponding text segments in source and target languages known as “translation units” (Simard, 2020). The size of translation memories and the quality of their contents are major impact factors crucial to the effectiveness of the translation memory system which uses them. Due to the importance of translation memories, there has been done lots of research work for building large TMs on worldwide scale, not just in individual countries (Steinberger et al. 2012).

What plays an important role for TM system is also the similarity evaluation between the input sentence to be translated and the source segment in the TM. The main task of TM system is to get a translation unit whose source segment is the most similar to the input sentence out of TM. There are two possible solutions in performing the task: One solution is to adopt an intelligent TM matching mechanism which is able to correctly calculate the similarity between the input sentence and the source segment in the translation memory. The other solution is to increase the size of TM by collecting translated resources as much as possible. No matter how intelligent the TM matching mechanism is, small-size TM cannot afford rich performance. Of course, the choice of TM matching method is important for improving the effectivity of TM system. But what is no less important than any TM matching method is to use a reasonable size TM. The larger the TM, the higher the possibility to get a translation unit whose source segment is very similar to the input sentence out of the TM. In general, the main value of a TM consists in the number of segments - its size. However, large TMs automatically lead to slow response times. A slow TM might actually slow down a translator, so that fast response time is an essential characteristic of any TM. Many research works have been reported to improve TM matching and retrieval, but the majority of those approaches were just evaluated on relatively small TMs. To our best knowledge, the largest TM tested so far in previous research works is the first five parts of the 2013 DGT-TM (which consisted of 305,324 segment pairs) used in (Weitz 2017) and the 2018 Volume 1 of the DGT-TM (which had 230,000 segment pairs) used in (Ranasinghe et al. 2020).

The main problem we are going to solve in this paper is to provide a TM retrieval mechanism to ensure real-time performance on very large TMs, e.g. with millions of segment pairs. We propose a TM retrieval method based on Vector Model (VM), which is widely used in information retrieval (IR), and implement our proposal using Lucene, an open source IR search engine. The rest of the paper is organized as follows: Section 2 briefly reviews previous research works related to TM matching and retrieval. Section 3 describes TM retrieval
method based on VM, and Section 4 presents experiment result. Finally, Section 5 discusses the result and draws a conclusion.

2 Previous Work on Translation Memory Matching and Retrieval

2.1 Research Work to Improve Translation Memory Matching

The mission of TM matching is to evaluate how similar the source segment in the TM is to the input sentence to be translated. Hence most of research work on TM matching focuses on how to calculate the similarity between the input sentence and the source segment in the TM.

(Planas and Furuse 2000) introduces edit distance based similarity vector whose coordinates refer to the levels of analysis of the segments. Their Multi-level Similar Segment Matching (MSSM) algorithm uses 3 different levels of data (surface words, lemmas, parts of speech (POS)) in a combined and uniform way.

There are studies for improving TM matching by segmenting source sentences. It is less likely for exact matches to be found in most text types, and even less so for complex sentences. MetaMorpho TM (Hodász and Pohl 2005) also divides sentences into smaller chunks. Moreover, it uses a multi-level linguistic features (surface form, lemma, and word class) to determine similarity between two source-language segments, especially for morphologically rich languages like Hungarian. The so-called ‘second generation’ TM system SIMILIS (Planas 2005) performs chunking to split sentences into syntagmas to allow sub-sentence matching. (Timonera and Mitkov 2015) suggests improving translation memory matching by performing clause splitting on the source segment as a pre-processing step for TM match retrieval, since clauses both contain a subject and a verb, hence a “complete thought”, and therefore clause matches are more likely to be in context and to be actually used by the translator.

(Vanallemeersch and Vandegehinst, 2014) also proposes a method which performs matching at level of syntactic trees. The authors notice that tree matching method is “prohibitively slow”. (Pekar and Mitkov 2007) proposes the ‘third-generation translation memory’ which introduces the concept of semantic matching. They employ syntactic and semantic analysis of segments stored in a TM to produce a generalized representation of segments which reduces equivalent lexical, syntactic and lexicosyntactic constructions into a single representation. Then, a retrieval mechanism operating on these generalized representations is used to search for useful previous translations in the TM.

(Chatzitheodorou 2015) presents an approach to match sentences having different words but the same meaning. They use NooJ to create paraphrases of Support Verb Constructions (SVC) of all source translation units to expand the fuzzy matching capabilities when searching in the translation memory.

(Ranasinghe et al. 2020) introduces a TM matching and retrieval method based on Universal Sentence Encoder. They argue that their method is an effective and efficient solution to replace edit distance based algorithms.

2.2 Research Work to Improve Translation Memory Matching

The mission of translation unit retrieval is to filter translation units out of TM which are to be matched against the input sentence. In general, the time consumed for translation unit retrieval is linear to the size of TM. Levenshtein distance, which is widely being used and one of the simplest means for TM matching, can be computed with dynamic programming in $O(mn)$ time, where $m$ is the length of the input sentence, and $n$ the length of the source segment of a translation unit in the translation memory. However, in case of a large TM with more than tens of millions of segment pairs, computing edit distance against the whole TM is too slow for real-time use. This is why the preliminary retrieval is necessary.

(Dandapat et al. 2012) uses an open-source IR engine SMART to retrieve a potential set of candidate sentences (likely to contain the closest match sentence) from the example base. Unigrams extracted from the sentences of the example-base are indexed using the language model and complete sentences are considered as retrievable units. They reported that finding a set of candidate sentences took only 0.3 seconds and 116 seconds, respectively, for 414 and 10,000 example input sentences given 20k and 250k sentence example base on a 3GHz Core 2 Duo machine with 4GB RAM. In order to find the closest matching sentences efficiently, (Dandapat et al. 2012) also proposes a heuristic-based grouping method which divides the example-base into bins based on
sentence length and considers only the segments which have comparable length to the length of the input sentence.

(Wäschle and Riezler 2015) uses MinHash signatures, an efficient way to estimate the similarity of two documents, to efficiently approximate the n-gram overlap of the input sentence and the source segment by representing each sentence as a set of n-grams in that n-gram overlap is a good predictor of TM match quality.

In order to reduce the search space size for Korean-Chinese TM retrieval, (Ryang 2018) builds a structured index using as features the sentence length and the sequence of Korean particles which is included in the sentence.

3 Vector Model-based Similarity Evaluation for Translation Memory Retrieval

3.1 Primary and Secondary Retrieval of Translation Memory

When retrieving from a large TM, it is common and reasonable to use the two-stage approach in which the TM system, firstly, filters candidates likely to be related to the input sentence for TM matching and then finds the most similar segments by fine-grained matching. The filtering is referred to the primary retrieval and the fine-grained finding is referred to the secondary retrieval. (Figure 1)

The primary retrieval is intended to filter translation units whose source segment is likely to be close matched with the input sentence. The secondary retrieval returns as reference translation the target segments of the translation units whose source segment is best matched with the input sentence. The main difference between the primary and secondary retrieval lies in the fact that the secondary retrieval uses a certain similarity threshold, \( \mu \), and the count of the secondary retrieval output should be much smaller than the primary one, because the secondary retrieval output is for human. The primary and secondary retrieval can be formulated respectively as follows:

\[
TM(S_0,K_1) = \arg\max_{\sum_{(S_i,T_i) \in TM} FMS_1(S_0,S_i) = K_1} \sum_{(S_i,T_i) \in TM} FMS_1(S_0,S_i)
\]

\[
TM_\mu(S_0,K_2) = \arg\max_{\sum_{(S_i,T_i) \in TM} FMS_2(S_0,S_i) \geq \mu} \sum_{(S_i,T_i) \in TM} FMS_2(S_0,S_i)
\]

where

- \( FMS_1(S_0,S_i) \): similarity score of the input sentence \( S_0 \) and the source segment \( S_i \), used in the primary retrieval
- \( FMS_2(S_0,S_i) \): similarity score of the input sentence \( S_0 \) and the source segment \( S_i \), used in the secondary retrieval
- \( TM = \{(S_i,T_i)|i=1,\ldots,N\} \): Translation Memory
- \( (S_i,T_i) \): Translation Unit, \( S_i \): Source Segment, \( T_i \): Target Segment
- \( N \): the number of translation units in the translation memory
- \( K_1, K_2 \): the limit count of the primary/secondary retrieval output

One of the essential requirements which the similarity measure should meet for the primary retrieval of TM is to allow partial match. A useful solution to this requirement is to use vector model
by representing as vectors the input sentence and
the source segments in the translation memory. We
adopt vector model based similarity evaluation for
the primary retrieval of TM.

3.2 Primary and Secondary Retrieval of
Translation Memory

For the vector representation of the input sentence
and the source segments in the TM, we use word-
sentence relation matrix which is widely used in IR.
Let $W$ be the set of words occurring in the source
segments.

$$W = \{w_i | i = 1, T\}, T$: The total number of words
occurring in the source segments

Let $V$ be the word-sentence relation matrix. Then $V$ is a $N \times T$ dimensional matrix:

$$V = \begin{pmatrix}
    v_{11} & v_{12} & ... & v_{1T} \\
    v_{21} & v_{22} & ... & v_{2T} \\
    ... & ... & ... & ... \\
    v_{N1} & v_{N2} & ... & v_{NT}
\end{pmatrix}$$

The i-th row of $V$ is a vector representing the
source segment, $S_i$:

$$V_{S_i} = (v_{i1}, v_{i2}, ... , v_{iT})$$

$\nu_{ij}$: The weight indicating how important the word
$w_j \in W$ is for the source segment $S_i$

Let $U_{S_0}$ be the vector of the input sentence $S_0$:

$$U_{S_0} = (u_1, u_2, ... , u_T)$$

$\nu_j$: The weight indicating how important the word
$w_j \in W$ is for the input sentence $S_0$

Given two vectors, $V_{S_i}$ and $U_{S_0}$, the similarity
score of the input sentence $S_0$ and the source
segment $S_i$ used for the primary retrieval can be
defined as follows:

$$FMS_1(S_0, S_i) = \frac{U_{S_0} \cdot V_{S_i}}{\|U_{S_0}\| \|V_{S_i}\|}$$

$U_{S_0} \cdot V_{S_i}$: Dot product of two vectors, $U_{S_0}$ and $V_{S_i}$

$\|U_{S_0}\|$: Euclidean norm of the vector $U_{S_0}$

$\|V_{S_i}\|$: Euclidean norm of the vector $V_{S_i}$

We suggest using TF-IDF weight of the words,
which is commonly used feature for IR. But there
is one problem in using TF-IDF weight for TM
retrieval.

In IR, the length of a query is very short than
documents. However, in case of TM retrieval, the
lengths of the input sentence and the source
segment, two objects to be compared, don’t make
such contrastive difference as in the relationship
between the query and document in IR. It can
rather be assumed that the length of the input
sentence is similar to the source segment in the TM.
Even when the length of the input sentence is short
than the source segment, as in IR, if a word occurs
only once in the input sentence, it is not true that a
source segment, in which that word occurs two or
three times, is more similar to the input sentence
than any other source segment in which that word
occurs once. When a word occurs twice in the
input sentence, it can be assumed that a source
segment, in which that word occurs twice, is more
similar than any other source segment in which that
word occurs once. Based on this consideration, we
define the elements of the vectors $U_{S_0}$ and $V_{S_i}$ as:

$$\nu_{ij} = \min(\text{tf}(w_j, S_0), \text{tf}(w_j, S_i))$$

$$\nu_j = \begin{cases}
    \text{idf}(w_j), & w_j \in S_0 \\
    0 , & w_j \not\in S_0
\end{cases}$$

Consequently, the similarity score of the input
sentence $S_0$ and the source segment $S_i$ becomes:

$$FMS_1(S_0, S_i) = \frac{\sum_{w_j \in S_0} \min\{\text{tf}(w_j, S_0), \text{tf}(w_j, S_i)\} \times \text{idf}(w_j)}{\|U_{S_0}\| \|V_{S_i}\|}$$

In the calculation of the above similarity score
function, the elimination of the term $\|U_{S_0}\|$ from
the denominator doesn’t affect the final result. So
the practical similarity score function can be
written as:

$$FMS_1(S_0, S_i) = \frac{\sum_{w_j \in S_0} \min\{\text{tf}(w_j, S_0), \text{tf}(w_j, S_i)\} \times \text{idf}(w_j)}{\|V_{S_i}\|}$$

3.3 Semantic Similarity for the Primary
Retrieval of TM

There are many previous research works taking
into account semantic similarity for TM matching.
For example, given two sentences, “What is the
actual aim of this practice?” and “What is the real
goal of this mission?”

It is possible to judge that
these two sentences are very similar, based on the
linguistic analysis that the words “actual” and
g“goal” are similar to the words “real” and “aim,”
respectively. When implementing two-stage TM
retrieval which consists of primary and secondary
retrieval for a large TM, it is very important to
ensure that the output of the primary retrieval
might contain the segments likely to be
semantically similar to the input sentence for any
semantic similarity measure to be applied at the secondary retrieval stage. We are going to use linguistic knowledge like synonym for accommodating semantic similarity evaluation in the primary retrieval of TM.

Our solution to evaluate semantic similarity taking into account the synonym knowledge in the primary retrieval of TM, is to change the input sentence \( S_0 \) into a pseudo sentence \( S'_0 \) which includes all the words of \( S_0 \) and also their synonym words, and then calculate the similarity score of the pseudo sentence \( S'_0 \) and the source segments of TM. The pseudo expansion of the input sentence and the similarity score calculation is trivial since the vector representation is based on TF-IDF weights. The similarity score of the pseudo sentence \( S'_0 \) and the source segment \( S_j \) is:

\[
FMS_j(S'_0, S_j) = \frac{\sum_{w_j \in S'_0} \min(tf(w_j, S'_0), tf(w_j, S_j)) \times idf(w_j)}{\|V_{S_j}\|}
\]

where

\[
tf(w_j, S'_0) = \begin{cases} 
\alpha, & w_j \in S_0 \\
0, & w_j \notin S_0 \land w_j \in SYN(w_k) \\
\end{cases}
\]

\[
SYN(S_0) = \bigcup_{w_j \in S'_0} SYN(w_j). SYN(w_k): The set of synonym words of \( w_l \)
\]

In the above expression, \( \alpha \) is a real number between 0 and 1, which is introduced as a weight of the synonym word added into the pseudo input sentence \( S'_0 \).

The knowledge database for synonym are not always available for every language, and even if available, they are qualitatively and quantitatively different from each other. For English, WordNet developed by Princeton University is a useful knowledge database for finding synonym.

As a matter of fact, it is not quite easy to find correctly the synonym of any word in the input sentence. To speed up the primary retrieval on a large TM while avoiding complex linguistic analysis, we establish the following principle for building synonym dictionary which will be used in the similarity evaluation for TM retrieval.

First, for any word \( w_i \) which has only one part-of-speech (POS), its all synonym words will be included in the synonym dictionary \( SYN(W) \).

Second, when the word \( w_i \) has several POSes, only if \( w_i \) doesn’t have verb POS, its synonym words will be included in the synonym dictionary \( SYN(W) \).

Third, if the word \( w_i \) has both general synonym and special synonym, only the general synonym words with more high frequency will be included in the synonym dictionary \( SYN(W) \).

Our principles are based on the linguistic consideration that the synonym of any word can be discussed only when its POS is determined, that there exist two categories of synonym, absolute synonym and relative synonym, and that there are also general synonym and special synonym in terms of use frequency.

We don’t use synonym of multi-POS words with verb POS, because a verb is the core of the statement unit which can determine the meaning of a sentence from a linguistic point of view and linguistic analysis like POS tagging is not applied in the primary retrieval of TM.

According to our analysis on WordNet 3.0\(^1\), it has a total of 117,659 senses with 147,306 words related to each other. Among those words, there are 49,754 words which does not have any synonym at all. Using above-mentioned principles for synonym selection, we selected 36,185 senses with 90,258 words related to each other to build an English synonym dictionary for TM retrieval.

4 Experimental Result

We use Lucene, an open source IR engine in Java, to implement the TM retrieval system using the vector model based similarity evaluation we proposed. As the data structure of a translation unit of TM, the Document class of Lucene is used which has two fields corresponding to the source and target segment of TM, respectively. The version number of Lucene used is 8.5.1. Levenshtein Distance based similarity score is applied for the secondary retrieval of TM. The TM used for the evaluation of the proposed TM retrieval system is an English-to-Korean TM which is made of about 5,000,000 English segments and their automatic Korean translation by English-to-Korean machine translator “Ryonngamsan” 2.0. In the experiment, we use parameter settings for the primary search of TM such that \( K_1 = 100 \), and \( \alpha = 0.5 \). All measurement was carried out on a desktop PC with

---

\(^1\) (http://wordnetcode.princeton.edu/3.0/WNprolog-3.0.tar.gz)
Intel® Core™ i3-3240 CPU @ 3.40GHz and 2GB of RAM. The operating system installed is Windows 7, 64bit.

4.1 Evaluation Method of Retrieval Performance

First of all, the retrieval performance on the English-to-Korean TM using Lucene can be evaluated with the retrieval time on varying size of TM. We randomly selected 1,000 sentences which are not included in the English-to-Korean TM, and then measured the total time consumed for retrieving all those sentences on different size of TM. The time consumed for retrieving was measured 5 times, and the fastest, slowest and averaged time were all recorded. Next, the relevance of retrieval result was automatically tested. Finally, we compare the retrieving performance of our proposal with the retrieving performance when using MongoDB’s full text search API.

4.2 Result

– Relation between the size of TM and the retrieving time

Figure 2 shows the relation between the size of TM and the retrieving time.

As the size of TM increases, so does the retrieving time on the TM.

– Relation between the length of the input sentence and the retrieving time

We also investigate the influence of the length of the input sentence on the retrieving time on TM.

<table>
<thead>
<tr>
<th>Size of TM (×10,000)</th>
<th>Retrieving time (milliseconds) (Number of Measurement)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>50</td>
<td>14,289</td>
</tr>
<tr>
<td>100</td>
<td>25,506</td>
</tr>
<tr>
<td>150</td>
<td>33,215</td>
</tr>
<tr>
<td>200</td>
<td>42,523</td>
</tr>
<tr>
<td>250</td>
<td>47,191</td>
</tr>
<tr>
<td>300</td>
<td>59,027</td>
</tr>
<tr>
<td>350</td>
<td>72,304</td>
</tr>
<tr>
<td>400</td>
<td>78,106</td>
</tr>
<tr>
<td>450</td>
<td>85,019</td>
</tr>
<tr>
<td>500</td>
<td>96,876</td>
</tr>
</tbody>
</table>

Table 1: Retrieving time according to the size of TM

Figure 2: Retrieving time according to the size of TM

For 1,000 input sentences being tested, the retrieving time for each sentence on the largest TM with 5,000,000 segments was measured and averaged according to the length of those sentences. Figure 3 and Figure 4 show the sentence length-frequency distribution on the test sentences and the average retrieving time according to the sentence length, respectively.

The result shows that the longer the input sentence, the longer its retrieving time of TM.

Figure 3: Sentence frequency according to its length

Figure 4: Average retrieving time according to the sentence length
Relevance of the primary retrieval result of TM

For evaluating the relevance of the primary retrieval result of TM, we checked the ranking result of the primary retrieval when retrieving 1,000 English sentences randomly selected from the largest TM of 5,000,000 segments. According to an automatic checking of the ranking result, the translation unit whose source segment is the input sentence ranked at the first place all the time. This implies that the proposed primary retrieval of TM is relevant for exact match of TM.

The relevance of the primary retrieval result for the sentences which is not included in the TM is impossible to automatically evaluate, and is also related to the secondary retrieval of TM. We did a small manual test but the result was not fully reliable, so we didn’t present the result here.

Comparison with the retrieving performance of TM using MongoDB

MongoDB, a NoSQL database management system, supports textual data indexing and searching which allows partial matching. For comparison with our proposal, we implemented a TM retrieval module using the full text search API of MongoDB, and evaluated its performance on a desktop PC with Intel® Core™ i7-7700 CPU @ 3.6 GHz and 16 GB of RAM. The version of MongoDB used is 4.4. It took about 18 minutes to insert into the MongoDB collection the English-to-Korean TM of 5 million segment pairs. It also took about 5 minutes to index the source language field and about 1 hour and 38 minutes to retrieve a set of candidate sentences for the same 1,000 English sentences as in the previous experiment. The size of the set of candidate sentences was limited to 10. By automatically checking the relevance of the retrieval result, the translation unit whose source segment is the input sentence ranked at the first place all the time, too. Obviously, the retrieving speed when using Lucene is incomparably superior to when using MongoDB.

5 Conclusion and Future Work

Through a series of experiments on the primary retrieval of English-to-Korean TM using vector model based similarity evaluation, we conclude that:

- The time and space complexity of indexing the TM increases linear to the size of the TM. The indexing time consumed for a large-scale English-to-Korean TM with about 5,000,000 segments is about 5 minutes, and the indexed data size is 1.84 GB with an increase of about 29 % compared to the text file size of the TM.
- The time complexity of the primary retrieval of TM increases linear to the size of the TM and the length of the input length. The fact that the retrieving time of TM is in linear proportion to the size of the TM and the length of the input sentence fully accords with Lucene’s inverted indexing principle and the ranking process of our vector model based similarity evaluation.
- When there is a translation unit whose source segment is the same as the input sentence, the translation unit ranks at the first place in the primary retrieval result of TM. The automatic checking result of the source segments included in the TM shows that Lucene is an effective means for exact match, as well as fuzzy matching.

The effect of the vector model based similarity evaluation for the primary retrieval of TM wholly depends on the correctness of the morphological analysis and the richness of the synonym knowledge. Since the difficulty of the morphological analysis and the availability of the synonym knowledge like WordNet is all different for each language, we plan to do more research work on these aspects. Furthermore, we also plan to evaluate more comprehensively the relevance of the primary retrieval of TM.

References


Now, It’s Personal : The Need for Personalized Word Sense Disambiguation

Milton King
Faculty of Computer Science
University of New Brunswick
milton.king@unb.ca

Paul Cook
Faculty of Computer Science
University of New Brunswick
paul.cook@unb.ca

Abstract
Authors of text tend to predominantly use a single sense for a lemma that can differ among different authors. This might not be captured with an author-agnostic word sense disambiguation (WSD) model that was trained on multiple authors. Our work finds that WordNet’s first senses, the predominant senses of our dataset’s genre, and the predominant senses of an author can all be different and therefore, author-agnostic models could perform well over the entire dataset, but poorly on individual authors. In this work, we explore methods for personalizing WSD models by tailoring existing state-of-the-art models toward an individual by exploiting the author’s sense distributions. We propose a novel WSD dataset and show that personalizing a WSD system with knowledge of an author’s sense distributions or predominant senses can greatly increase its performance.

1 Introduction
Authors of text tend to predominantly use a single sense for a lemma that can differ among different authors (Gella et al., 2014). This might not be captured with an author-agnostic word sense disambiguation (WSD) model that is trained on multiple authors. Our work finds that WordNet’s first senses, the predominant senses of our dataset’s genre, and the predominant senses of an author can all be different and therefore, author-agnostic models could perform well over the entire dataset, but poorly on individual authors. Ideally, each author would have access to a personalized WSD model, which is a model that was tailored toward that individual. In this work, we explore methods for personalizing WSD models by tailoring existing state-of-the-art models toward an individual by exploiting the author’s sense distributions. We evaluate our models on our proposed dataset which contains 1586 sense-annotated instances for 11 lemmas across 36 authors. Our evaluation includes metrics that focus on the performance of models with respect to the authors that they perform poorly on, which highlights the potential gain for individual authors.

The most similar work to this is presented in Gella et al. (2014), which created a dataset that contains sense-annotated instances of tweets for a list of authors. We differ from them by including more annotated instances from a single author and by using text from blog posts, which do not have a limitation on the length of text, unlike text from tweets. We evaluate different WSD models, including a state-of-the-art model (SensEmBERT) (Scarlini et al., 2020), which we extend with personalization techniques to achieve our best scores. We personalize SensEmBERT with knowledge of an author’s sense distributions or predominant senses, which outperform author-agnostic WSD systems. Our work also shows that the use of author-specific sense distributions outperforms the use of genre-specific senses. In this first work on personalized WSD, we do not automatically learn the sense distributions of an author, but instead, we use the author’s true sense distributions to demonstrate the importance of learning author-level sense distributions when considering personalized WSD.

2 Related Work
Two common frameworks for WSD systems include knowledge-based, which utilizes information contained in a sense inventory, and supervised, which involves training on annotated instances. In this work, we focus on knowledge-based models because they do not require annotated instances and they are able to work well on less frequent lemmas and senses (Scarlini et al., 2020). Many modern knowledge-based methods extend the simplified Lesk method (Kilgarriff and Rosenzweig, 2000), which involves classifying a token with the sense that contains the most overlapping words in its defi-
nition with the context of the word being classified.
WSD methods that involved a Lesk-style approach
are Banerjee and Pedersen (2002), which included
looking at the definition of words that are similar
to the token being classified according to Word-
Net and Basile et al. (2014), which looked at the
definition of similar words but added a vector repre-
sentation of the target token — generated by a topic
model — into their similarity calculations. Topic
modeling is a probabilistic model that views docu-
ments as a distribution over topics and topics as
distribution over types (Blei et al., 2003). Topic
modeling was also applied to WSD in Boyd-Graber
et al. (2007) and Li et al. (2010).

The current knowledge-based model, that
achieves state-of-the-art performance on the uni-
ified WSD evaluation framework (Raganato et al.,
2017), creates an embedding for each sense of a
word by using BERT (Devlin et al., 2019) to embed
both the sentences in Wikipedia articles related to
that sense — determined by words in the sense’s
WordNet synset — and the gloss of the sense (Scar-
lini et al., 2020). The two vectors are then con-
catenated to make the final sense embedding. At
runtime, the context of the target word is embed-
ed using BERT and is compared to the previously
described sense embeddings to determine which
sense is assigned. SensEmBERT works well on rare
lemmas and senses (Scarlini et al., 2020), which is
important to consider for our experiments that
contain author-specific text since authors tend to
favour a single sense for a word (Gella et al., 2014),
which might not be the predominant sense of a do-
main. Therefore, if a model performs poorly on
rare senses according to the sense inventory, then
it might perform poorly on the favoured sense of
the author. Likewise, if an author frequently uses
rare words — according to the sense inventory —
then a model that performs poorly on rare words
would perform poorly for the author. The ability to
perform well on all authors and demographics is a
way to evaluate the fairness of a model (Ethayarajh
and Jurafsky, 2020; Hashimoto et al., 2018). In this
work, we consider the fairness of our models by
focusing on authors that they perform poorly on.

3 Data Statement

In this section, we discuss the properties of our
dataset, while following the proposed schema from
Bender and Friedman (2018).

3.1 Data selection

We collect all text from blogs of the top 50 authors
that contain the most tokens from the corpus that
was originally presented in Schler et al. (2006).
The original corpus consists of English blog posts
from 19,320 authors. There were some authors
who have blog posts that are copied from other
authors and therefore, we do not consider these au-
thors in the top 50. We consider the top 50 authors
to ensure that each author possesses enough text to
allow the ability to study the potential benefits of
using text from the author for personalizing a WSD
model. For selecting lemmas, we first consider all
20 nouns from Gella et al. (2014). We consider the
top 10 authors — authors that have the $10^{th}$ highest
frequency of a lemma — to ensure that there is
text from multiple authors for each lemma to study
the effects of personalization on a per-lemma basis.
From this group of lemmas, we retain all lemmas
that have been used 20 or more times by the author
who has used this lemma the $10^{th}$ most frequently.
We selected the cutoff of 20, because there is not a
large possibility that all instances from an author
will be usable due to them not being a noun or not
representing a sense from our chosen sense inven-
tory. The group of lemmas that we include moving
forward will be referred to as SHORT LIST.

For each lemma in SHORT LIST, we randomly
sample approximately 10 sentences that contain
the lemma from 10 authors and manually assign
the lemma a sense ourselves.¹ We then use this
annotated subset to perform two different analy-
ses that focus on quantifying the diversity of the
senses for a lemma. The first analysis involves
calculating the predominant sense of each lemma
for each author and then finding the most frequent
sense among the author-level predominant senses,
which we call the grand sense. We then calculate
the percent of author-level predominant senses for
a given lemma that are not the grand sense. The
second analysis calculates the number of assigned
senses that are not the grand sense. Both types
of analysis assist in showing which lemmas have
senses that vary among authors and therefore, they
might benefit from a personalized model which is
the main focus of this work. Lemmas that score
low on these metrics could be ideal for models
that predict the predominant sense, but would most
likely not benefit from an author-level model. We
originally wanted the top 10 lemmas that scored

¹We are native English speakers.
Table 1: List of lemmas and their frequency for the author who uses the lemma the 10th most frequently (10th MF). The percent of author-level predominant senses and token senses that are not the grand sense, represented by Predom and Token, respectively (higher values indicate more diversity). The number of WordNet senses for each lemma under the label #Senses is also shown.

<table>
<thead>
<tr>
<th>Lemma</th>
<th>10th MF</th>
<th>Predom</th>
<th>Token</th>
<th># Senses</th>
</tr>
</thead>
<tbody>
<tr>
<td>form</td>
<td>54</td>
<td>0.60</td>
<td>0.69</td>
<td>16</td>
</tr>
<tr>
<td>position</td>
<td>40</td>
<td>0.60</td>
<td>0.74</td>
<td>16</td>
</tr>
<tr>
<td>degree</td>
<td>27</td>
<td>0.56</td>
<td>0.57</td>
<td>7</td>
</tr>
<tr>
<td>sign</td>
<td>77</td>
<td>0.50</td>
<td>0.62</td>
<td>11</td>
</tr>
<tr>
<td>track</td>
<td>36</td>
<td>0.44</td>
<td>0.67</td>
<td>12</td>
</tr>
<tr>
<td>paper</td>
<td>54</td>
<td>0.44</td>
<td>0.57</td>
<td>7</td>
</tr>
<tr>
<td>deal</td>
<td>75</td>
<td>0.40</td>
<td>0.44</td>
<td>9</td>
</tr>
<tr>
<td>field</td>
<td>30</td>
<td>0.30</td>
<td>0.43</td>
<td>17</td>
</tr>
<tr>
<td>case</td>
<td>97</td>
<td>0.22</td>
<td>0.47</td>
<td>19</td>
</tr>
<tr>
<td>charge</td>
<td>36</td>
<td>0.22</td>
<td>0.34</td>
<td>15</td>
</tr>
<tr>
<td>rule</td>
<td>43</td>
<td>0.22</td>
<td>0.27</td>
<td>12</td>
</tr>
</tbody>
</table>

the highest when comparing predominant senses with the grand sense, but we received a tie for the lemmas that scored 9th, 10th, and 11th. We remove all lemmas from SHORT_LIST that scored less than 0.22 on the percent of predominant senses that are not the grand sense, which was the score for the lemmas with the 9th, 10th, and 11th highest values. Table 1 shows our final 11 lemmas with their frequency for the top 10 authors, and their two diversity metrics.

Additional preprocessing was applied to the text from authors and the annotated instances, including the replacing of tokens that contain URL identifiers (www, html, https, etc.) with the token urlLink and the removal of what appear to be artifacts of text encoding, such as \texttt{xx}.

For each of the 11 lemmas in the dataset, we gather the top 10 authors that used that lemma most frequently and gather all sentences from them that contain that lemma tagged as a noun by a part-of-speech tagger (Qi et al., 2020). This results in 36 authors and a total of 1607 instances across all lemmas. We manually scan through all instances ourselves and remove all instances that were incorrectly tagged as nouns.

3.2 Annotation

In this work, we use WordNet (Miller, 1995) as the sense inventory due to its popularity among WSD tasks (Raganato et al., 2017). We show the number of WordNet senses for each lemma in Table 1, which ranges from 7 for degree to 19 for case.

We used Amazon Mechanical Turk — a common crowd-sourcing site — to annotate the instances of each lemma. We divided the instances into groups of 5 — known as HITs — and ensure all 5 instances belong to the same lemma. Each instance in a HIT is presented to the annotator, known as a turker, with a piece of text that contains a single target token written in bold for each text. Each instance contains up to 20 tokens before the target token and 20 tokens after the target token, which can cross sentences but does not cross blog posts. This provides the turker with more context than only looking at a single sentence, which can assist their annotations. The turkers are asked to select the sense that best applies to the target token from a list of the WordNet senses for the lemma of the target token or they can select I cannot assign a sense. There was space available for feedback for each instance, where turkers can write the reason that they cannot assign a sense or provide general feedback.

It is possible that a token can exhibit multiple senses (Erk et al., 2009), but our dataset will only consider one sense as the ground truth for each instance. Therefore, following Chkolvski and Mihalcea (2002) and Pradhan et al. (2007), a turker can only select one sense for any instance. Each HIT was annotated by 10 turkers. An example of the task assigned to turkers is seen in Figure 1.

3.2.1 Annotators

For a turker to be eligible to annotate the HITs, they need to be 19 years of age, speak English as a first language, live in Canada or United States, and have a previous HIT acceptance rate of 98%. We paid the turkers between $0.05 and $0.10 per HIT, which is competitive with other sense annotation tasks (Akkaya et al., 2010; Hong and Baker, 2011; Rumshisky, 2011; Passonneau and Carpenter, 2014). Our work consists of 185 annotators producing a total of 14,607 annotations and 137 instances of feedback.

Some turkers may provide poor annotations and therefore an initial pass over the annotations can help identify these turkers (Gella et al., 2014). Gella et al. (2014) included a gold-standard instance within each HIT and disregarded all annotations from turkers that performed poorly on these gold-standard instances. Instead of providing a gold-standard instance within each HIT, we compare each turker’s annotations against a majority vote. We avoid the use of a gold standard because we did not want to mix text types, i.e., blog posts and WordNet, and because not all senses in WordNet have an example, and those that do have
examples do not always contain the target lemma. We do this by performing an initial pass over the annotations to calculate the majority vote for each instance and then calculate each annotator’s accuracy with the majority vote. Annotations from any turker that scored less than 50% agreement with the majority vote are removed from consideration, leaving 162 turkers in the dataset. We perform a second pass through the dataset and calculate the majority vote for each instance and remove 7 instances, which were assigned I cannot assign a sense as the most frequent label and 2 instances where there was a tie for the most frequent label. We randomly annotated 86 instances and our annotations agreed with the turkers’ majority vote 85% of the time.

3.3 Speech Situation
All text was originally obtained from downloading all accessible blogs from blogger.com on a single day in August of 2004 (Schler et al., 2006). The number of instances per author ranges from 3 to 152 with a mean of 44 and a median of 31 instances per author. The age of the authors range from 17 to 48 years with a mean of 30 and a median of 27. The sex of the authors is disproportionate, with 10 females and 26 males.

3.4 Speaker Demographic
All text in the original corpus was English, although there was non-English text in the blogs, which was removed by Schler et al. (2006).

3.5 Dataset Analysis
The final dataset consists of 11 lemmas and 1586 annotated instances. The dataset consists of multi-sentence instances that were annotated by the turkers. We did this to maintain consistency with the text being annotated by turkers and the text being used by WSD models. Table 2 shows the number of instances per author for each lemma ranges from 93 for charge to 192 for paper with an average of 144. The number of assigned senses ranges from 4 for deal to 13 for field with an average number of 8.5, known as sense ambiguity (Jurgens, 2014). The sense ambiguity is a metric that can be used to measure the difficulty of a WSD dataset. The dataset’s sense ambiguity of 8.5 is among the higher values of the datasets in the collection from Raganato et al. (2017) and the dataset from Gella et al. (2014), which range from 4.9 to 8.9.

Figure 2 shows the sense distributions for four authors with respect to the lemma deal and shows how authors can use the same lemma differently and the potential benefits of tailoring models toward an individual author. Specifically, each author in Figure 2 has a different predominant sense for the lemma deal and only Author.0 shares their predominant sense with the predominant sense across all authors.

4 Methods
In this section, we discuss the WSD methods that we evaluate on the dataset. This includes predominant sense baselines, a state-of-the-art method (SensEmBERT), and our proposed personalized models.

4.1 Baselines
We apply three WSD baselines, which include always predicting the predominant sense for each lemma. The predominant sense is calculated via Mordecaffe/Personalized_WSD_Dataset.
Table 2: The number of instances and assigned senses for each lemma.

<table>
<thead>
<tr>
<th>Lemma</th>
<th># instances</th>
<th># senses assigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper</td>
<td>192</td>
<td>7</td>
</tr>
<tr>
<td>Position</td>
<td>176</td>
<td>12</td>
</tr>
<tr>
<td>Sign</td>
<td>163</td>
<td>9</td>
</tr>
<tr>
<td>Form</td>
<td>156</td>
<td>10</td>
</tr>
<tr>
<td>Case</td>
<td>154</td>
<td>10</td>
</tr>
<tr>
<td>Degree</td>
<td>146</td>
<td>6</td>
</tr>
<tr>
<td>Track</td>
<td>146</td>
<td>8</td>
</tr>
<tr>
<td>Deal</td>
<td>140</td>
<td>4</td>
</tr>
<tr>
<td>Field</td>
<td>121</td>
<td>13</td>
</tr>
<tr>
<td>Rule</td>
<td>99</td>
<td>6</td>
</tr>
<tr>
<td>Charge</td>
<td>93</td>
<td>8</td>
</tr>
<tr>
<td>Average</td>
<td>144</td>
<td>8.5</td>
</tr>
</tbody>
</table>

Figure 2: Sense distributions for four authors for the lemma deal. The average distribution across all authors is in black.

WordNet (WORDNET), the predominant sense of the dataset (DATASET_PREDOM), and the predominant sense for each author (AUTHOR_PREDOM).

4.2 SensEmBERT

This method is an unsupervised method that has achieved state-of-the-art results on the English datasets from Raganato et al. (2017) and can outperform supervised WSD models on less frequent lemmas and senses (Scarlini et al., 2020). It uses sense embeddings of a noun by embedding text from Wikipedia articles related to the noun and concatenating it with an embedding of text from the noun’s BabelNet entry. The embedding process of a token is done by averaging BERT embeddings of each considered token. A BERT embedding is calculated by summing the last four layers of BERT after using the target token in context as input.

SensEmBERT assigns a token’s sense by embedding the token with BERT and concatenating this embedding onto itself to double the vector length. Cosine similarity is calculated between this vector and all sense embeddings for all possible senses for the lemma of the target token. The sense that has the highest similarity is assigned as the target token’s sense.

4.3 Personalizing SensEmBERT

We extend SensEmBERT by using text from the author to tailor the model to them by assuming knowledge about the author’s sense distributions. These methods are used to explore the benefits of personalized WSD systems and the potential gains of learning the sense distributions of an author. We discuss these types of methods in this subsection.

4.3.1 SEBERT_PERS

In this method, we exploit the Zipfian distribution that sense frequencies tend to exhibit (Kilgarriff, 2004). Specifically, the weights for each sense that are outputted by SensEmBERT are ranked and the final score for a sense is calculated by the inverse rank of the sense multiplied by the probability of this sense given an author (calculated by the author’s gold standard sense distributions) as seen in Equation 1. The sense that results in the highest score is assigned as the sense of the target token.

$$weight(sense) = \frac{1}{rank} \times p(sense|author)$$ (1)

4.3.2 SEBERT_PERS_PREDOM

The author-level sense distribution of a lemma could be difficult to automatically estimate and it might be easier to estimate the author-level predominant sense. Therefore, we assume knowledge of only the author-level predominant sense of a lemma for this method by using the author’s gold standard predominant sense. Specifically, we assign the sense that was given the most weight according to SensEmBERT if the predominant sense of the author is not among the top k ranked senses. If the predominant sense is in the top k ranked senses, we assign the predominant sense. We refer to this k value as the override rank and it is a hyperparameter that needs to be tuned. We also explore the use of the predominant senses from the dataset and WordNet; we refer to these methods as SEBERT_DATASET_PREDOM and SEBERT_WORDNET_PREDOM, respectively.

5 Experimental Results

In this section, we evaluate our different models. We first explore the tuning of the override rank for PREDOM-based methods. We then evaluate our
models using average accuracy across all authors and for individual authors.

5.1 Tuning SEBERT_PERS_PREDOM
Unfortunately, due to the relatively small size of our dataset, we are unable to use a held-out subset of the data for model tuning. Therefore, we show the performance of the PRE-DOM-based models using different override ranks. Figure 3 shows that SEBERT_PERS_PREDOM and SEBERT_DATASET_PREDOM outperform the SensEmBERT approach for any of the tested override rank values except for SEBERT_DATASET with an override rank of 6. This finding eliminates the need to fine-tune this model, since any value between 2 and 5, inclusively, works reasonably well. Increasing the override rank results in the model that uses WordNet first-senses to perform worse, which suggests that the authors’ predominant senses do not align with WordNet’s first senses. We do not consider SEBERTWORDNET_PREDOM for the remaining experiments due to its poor performance and we use an override rank value of 4 for SEBERT_PERS_PREDOM and SEBERT_DATASET_PREDOM.

5.2 Overall Accuracy
In this subsection, we discuss the results of each method on the entire dataset. We evaluated each model using accuracy across instances, average accuracy across lemmas, and average accuracy across authors. We evaluated using these three metrics to eliminate the issue of having non-uniform distributions of instances in the dataset. For example, the number of instances per author ranges from 3 to 152 and, therefore, we would like to weigh each author equally in the case of accuracy across authors, instead of favouring models that only perform well on authors with more instances in the dataset.

Table 3 shows the scores for each method across the different evaluations. The three predominant sense baselines’ performances scored in the expected order, such that using the predominant sense of the author outperforms using the predominant sense of the dataset, which outperforms using the first sense from WordNet. SensEmBERT outperformed all other baselines. The inclusion of the author’s sense distribution in SEBERT_PERS and their predominant sense in SEBERT_PERS_PREDOM both outperform SensEmBERT. SEBERT_PERS achieves the highest score of 0.738 in terms of average accuracy across all authors, which is an absolute improvement of 0.127 above SensEmBERT. The use of the dataset-level sense distributions in SEBERT_DATASET and predominant senses in SEBERT_DATASET_PREDOM outperform SensEmBERT but does not outperform SEBERT_PERS and SEBERT_PERS_PREDOM, which supports the importance of using author-specific data. Interestingly, SEBERT_DATASET_PREDOM outperforms SEBERT_DATASET for average accuracy across authors. These findings indicate that SensEmBERT can be improved through personalization by incorporating information about author-level sense distributions or predominant senses.

5.3 Author-level Performance
In this subsection, we consider the models’ performances on individual authors and observe the lower bound of each model’s score with respect to the authors. By observing the lower bound of each model’s performance with respect to the authors, we can observe the fairness of the models.

Figure 4 shows the performance of our two personalized models (SEBERT_PERS and SEBERT_PERS_PREDOM) and the SensEmBERT
Figure 4: Author-level performances for the two personalized methods and SensEmBERT.

Figure 5: Performance of methods on the $x$ authors that they achieve the lowest accuracy on. The authors are sorted in ascending order with respect to the accuracy of SensEmBERT. It shows that authors that SensEmBERT performs below average on with respect to accuracy across authors (i.e. 0.611) often receive the largest boost in performance from personalized models. This could be due to those individuals having different writing styles as compared with text that SensEmBERT was trained on, which is an interesting topic for further exploration. The authors that achieve approximately 0.70 or greater for SensEmBERT can have their performance hindered by personalization, although, often not by a large amount. SEBERT_PERS usually outperforms SEBERT_PERS_PREDOM for a given author.

One of the main pillars of our work is to provide personalized models that work well for authors that scored poorly with conventional non-personalized models. Therefore, we would ideally want models that do not perform poorly on any single author, which we can evaluate by averaging the accuracy over the authors that each model achieves the lowest accuracy on. In Figure 5, we evaluate our models on the $x$ authors that each model achieves their lowest accuracy on for values of $x$ ranging from 1 author to all 36 authors. It shows that SensEmBERT achieves 0.25 on its worst author, while SEBERT_PERS achieves 0.46 on its worst author. Furthermore, SEBERT_PERS, SEBERT_PERS_PREDOM, and SEBERT_DATASET_PREDOM always outperform SensEmBERT for every value of $x$ in the $x$ worst authors evaluation, with SEBERT_PERS always achieving the highest score. This finding demonstrates that Personalized WSD models such as SEBERT_PERS and SEBERT_PERS_PREDOM are more fair than non-personalized models (SensEmBERT).

6 Conclusions

In this work, we proposed a novel dataset for personalized WSD and showed that sense distributions and predominant senses of an author can be used to personalize an existing knowledge-based WSD model (SensEmBERT). Our experiments consistently show that models that consider author-specific sense distributions (SEBERT_PERS) or predominant senses (SEBERT_PERS_PREDOM) can outperform models that do not consider any knowledge of sense distributions (SensEmBERT). Furthermore, we show that models that use author-level sense distributions or predominant senses outperform models that use genre-level sense distributions (SEBERT_DATASET) or predominant senses (SEBERT_DATASET_PREDOM). SensEmBERT achieved the highest accuracy across all authors with an absolute improvement of 0.127 above SensEmBERT. We further explore the fairness of our models by evaluating their accuracy on authors that they perform poorly on and showed that the lowest accuracy achieved by SEBERT_PERS on a single author is 0.21 above SensEmBERT’s lowest accuracy. This finding demonstrates that SEBERT_PERS is more fair than SensEmBERT, which indicates that personalization can produce more fair WSD systems. Our work shows the importance of learning sense distributions of individual authors for WSD and therefore, we plan on developing methods for learning an author’s sense distributions in future work similar to Pasini et al. (2020) and Bennett et al. (2016). Our personalized models could be learning topic-related content from the author to assist with their classification, therefore, an extension of this work could further explore this dataset with a focus on topic-related features.
7 Ethical Considerations

The involvement of turkers as annotators was reviewed and approved by the University of New Brunswick’s ethics committee. We selected the authors based on their amount of text available and therefore the distributions over sexes is not equal — 26 males and 10 females — and therefore groups should consider this when working with this dataset.

Acknowledgments

This work is financially supported by the Natural Sciences and Engineering Research Council of Canada and the University of New Brunswick.

References


Jordan L. Boyd-Graber, David M. Blei, and Xiaojin Zhu. 2007. A topic model for word sense disambiguation. In EMNLP-CoNLL.


Abstract

In this paper, we present work in progress aimed at the development of a new image dataset with annotated objects. The Multilingual Image Corpus consists of an ontology of visual objects (based on WordNet) and a collection of thematically related images annotated with segmentation masks and object classes. We identified 277 dominant classes and 1,037 parent and attribute classes, and grouped them into 10 thematic domains such as sport, medicine, education, food, security, etc. For the selected classes a large-scale web image search is being conducted in order to compile a substantial collection of high-quality copyright free images. The focus of the paper is the annotation protocol which we established to facilitate the annotation process: the Ontology of visual objects and the conventions for image selection and for object segmentation. The dataset is designed both for image classification and object detection and for semantic segmentation. In addition, the object annotations will be supplied with multilingual descriptions by using freely available wordnets.

1 Introduction

We are surrounded by information represented by text, images and video data in multimodal streams. One of the processing tasks for large multimodal data streams is automatic image description (image classification, object segmentation and classification), which is directly connected with the task of image search, as well as with the expansion of the scope for automatic question answering regarding images.

The goal of our project Multilingual Image Corpus (MIC 21) is to provide a large image dataset with annotated objects and object descriptions in (at least) 20 European languages. The

Multilingual Image Corpus consists of an ontology of visual objects (based on WordNet) and a collection of thematically related images whose objects are annotated with segmentation masks and labels describing the ontology classes. The dataset is designed both for image classification and object detection and for semantic segmentation.

The main contributions of our work are: a) the provision of large collection of high-quality copyright free images; b) the formulation of the Ontology of visual objects based on WordNet noun hierarchies; c) the precise manual correction of automatic object segmentation within the images and the annotation of object classes; and d) the association of objects and images with extended multilingual descriptions based on WordNet inter- and interlingual relations.

We have divided the annotation process into four main stages: a) definition of an ontology of visual objects; b) collection of appropriate images; c) automatic object segmentation; and d) manual correction of object segmentation and manual classification of objects. The annotation protocol includes the Ontology of visual objects and the conventions for image selection and for object segmentation.

The focus of the paper is the annotation protocol which is established to facilitate the manual annotation. We begin with a brief overview of the current state in the field in Section 2. Section 3 is dedicated to the description of the Ontology of visual objects. Dataset collection is described briefly in Section 4. Section 5 provides an outline of the annotation protocol. Finally, conclusions and future directions of our work are presented.

We will show how the presented image dataset benefits from WordNet: providing ontological representation of visual objects based on WordNet noun hierarchies; building interconnectivity of classes by means of the WordNet relations; and ensuring multilinguality by using freely available wordnets.
2 Related Work

There is a tradition already established in the image dataset collection and annotation; the available datasets show an increase both in the number of training images and in the number of object classes. CalTech-256 dataset consists of 30,607 images and covers 256 object categories\(^2\) (classes). The annotation includes bounding boxes, in which the objects are located, and object outlines provided by humans (Griffin et al., 2007). The categories are organised in a taxonomy grouping categories into animate and inanimate and other finer distinctions; for example, the category electronics is divided further into entertainment, computing, home, office and others.

The CalTech 101 Silhouettes\(^3\) dataset consists of 4,100 training samples, 2,264 validation samples and 2,307 test samples. The dataset is based on CalTech 101 image annotations. Each image in the CalTech 101 Silhouettes dataset includes a high-quality polygon outline of the primary object in the scene (Marlin et al., 2010).

The TinyImages dataset (Torralba et al., 2008) is a large dataset containing 80 million small images (32 x 32 pixels) automatically collected from the Internet using 53,464 nouns from WordNet as queries. The dataset is not available online since it has not been manually evaluated\(^4\).

The Scene Understanding (SUN)\(^5\) collection contains 899 categories and 130,519 images. SUN annotates scene types and the objects that commonly occur in them. There are 397 categories designed to evaluate numerous state-of-the-art algorithms for scene recognition (Xiao et al., 2010). The SUN Attribute\(^6\) dataset consists of 14,340 images from 717 scene categories, and each category is annotated with a taxonomy of 102 attributes (Patterson et al., 2014).

ModaNet\(^7\) is a dataset consisting of annotations of street fashion images. ModaNet provides multiple polygon annotations for each image. Each polygon is associated with a label from 13 meta fashion categories (bag, belt, footwear, outer, dress, etc.), where each meta category groups highly related categories to reduce the ambiguity in the annotation process (Zheng et al., 2018).

There are several datasets which have been widely used as a benchmark for object detection, semantic segmentation and classification tasks.

The PASCAL Visual Object Classes (VOC) 2012\(^8\) dataset contains 20 object categories including vehicles, household, animals, and others: airplane, bicycle, boat, etc. Each image has pixel-level segmentation annotations, bounding box annotations, and object class annotations (Everingham et al., 2010).

LabelMe is a dynamically developing dataset\(^9\) which contains hundreds of thousands of polygon annotations, thousands of static images and sequence frames with at least one labelled object (Russell et al., 2008). A particular feature of this collection is that it is being developed by users who can add images and categories and can annotated uploaded images. This option however may result in some level of inconsistency based on the decisions of the different users about the annotation protocol. The WordNet noun synonymous sets (synsets) are used to extend the categories, to avoid the inconsistency by means of manual editing and to unify the descriptions provided by different users.

One of the collections that set standards in the increase of datasets sizes is ImageNet\(^10\). The aim is for a the dataset with about 50 million cleanly labelled full resolution images (Deng et al., 2009). Another important feature of this dataset is that it uses WordNet noun hierarchies for image collection and labelling. ImageNet uses 21,841 synsets and contains 14,197,122 annotated images organised by the semantic hierarchy of WordNet (as of August 2014) (Russakovsky et al., 2015).

The taxonomic organisation of nouns in WordNet allows for using more abstract and fine-grained categories when describing objects. WordNet is a semantic network whose nodes host synonyms denoting different concepts and whose arcs, connecting the nodes, encode different types of relations (semantic: genus-kind, part-whole, etc.; extralinguistic: membership in a thematic domain; inter-language: translation equivalents). The idea for organising the lexicon of a given language into a (lexico-)semantic network was first executed in the Princeton WordNet (Miller et al., 1990). Some of the fundamental ideas on which the WordNet

\(^2\)https://www.kaggle.com/jessicali9530/caltech256
\(^3\)https://people.cs.umass.edu/~marlin/data.shtml
\(^4\)https://groups.csail.mit.edu/vision/TinyImages/
\(^5\)https://vision.princeton.edu/projects/2010/SUN/
\(^6\)https://cs.brown.edu/~gmpatter/sunattributes.html
\(^7\)https://github.com/eBay/modanet
\(^8\)http://host.robots.ox.ac.uk/pascal/VOC/
\(^9\)http://labelme.csail.mit.edu/Release3.0/
\(^10\)https://www.image-net.org
is based encompass: a) the use of a semantic network which embraces taxonomies, meronomies and non-hierarchical relations with clearly defined properties which allow for quick and easy automatic processing; b) a different organisation of the lexicon in comparison with the traditional dictionaries where words are ordered alphabetically and the links among semantically related words (such as between sister hyponyms, between a whole and its parts, etc.) are not explicitly presented (Miller, 1986).

The COCO (Microsoft Common Objects in Context) dataset (Lin et al., 2014) contains more than 328,000 images with manually annotated object instances (2.5 million)\(^{11}\). The dataset has had several releases since 2014 and it addresses object detection, segmentation, keypoint detection and captioning. The different parts of the dataset are annotated with bounding boxes (for object detection) and per instance segmentation masks with 80 object categories; natural language descriptions of the images; keypoints (17 possible keypoints, such as left eye, nose); per pixel segmentation masks with 91 stuff categories, such as grass, wall; full scene segmentation, with 80 thing categories (such as person, bicycle, elephant); dense pose – each labelled person is annotated with a mapping between image pixels and a template 3D model.

The image processing is generally classified as model based (using manually-labelled training data) and search based (using automatically collected training data). The search based approaches might include: effective learning mechanisms for matching a given query (Li and Fei-Fei, 2010); methods for automatic removing of noisy images (Hua and Li, 2015); frameworks combining discovering of multiple textual queries, filtering of noisy textual queries and noisy images (Anvari and Athitsos, 2019; Yao et al., 2020).

In the largest collection of datasets available on the internet, 1,455 image datasets are listed\(^ {12}\) (as of August 2021) provided with descriptions and links to the sources and related papers. Among them 134 datasets are designed for semantic segmentation; 104 – for image classification and 102 – for object detection. Ten datasets provide polygon annotations.

To summarise, the tendency in image annotation is from small training datasets to large-scale collections which require crowdsourcing in order to engage a large amount of human effort. Although the number and the diversity of image datasets is constantly expanding still there is a huge demand for more datasets in terms of variety of domains and object classes covered.

3 Ontology of Visual Objects

In current practice, WordNet is usually used in generating text queries for creation of search based image collections. A Visual Concept Ontology is proposed which organises visual concepts (objects or abstract notions that are typically depicted in photos) (Botorek et al., 2014). For the construction of Visual Concept Ontology over 400 “significant” noun synsets (that have at least 300 hyponyms) are extracted from WordNet, then synsets with very “general” meaning such as entity or thing were removed. This results in 14 top-level ontology classes, which are divided further into 90 more specific classes. Semantically similar synsets are merged into a common class and additional links are established between semantically related synsets such as roof and house.

We identified 10 thematic domains: Sport, Medicine, Arts, Education, Food, Transport, Clothing, Security, Indoors, Nature. The proposed Ontology includes concepts which are particular for these domains.

Following the strategy for category selection of the ImageNet we applied the rule for no overlapping between the classes: “for any synsets i and j, i is not an ancestor of j” (Deng et al., 2009). Mutually exclusive classes are also defined for other well-known datasets, for example for the COCO thing and stuff classes (Caesar et al., 2018). As it was pointed out, the mutual inclusion might lead to some inconsistencies. An example was given with the PASCAL Context (Mottaghi et al., 2014) classes bridge and footbridge, which are in a parent-child relation (Caesar et al., 2018). The parent term can replace the child term in some context, but not vice versa, thus: if two images are annotated as bridge and footbridge respectively, it will not be known whether the parent concept can refer also to the child concept or not.

The Ontology of visual objects has the following components:

**Classes** which can be represented by visual objects and correspond to the respective WordNet concepts. Among the classes we made a differ-
entiation between dominant classes and attribute (contextual) classes.

Each thematic domain is represented by several dominant classes, which show the main “players” within this domain differentiated by their type or their function. For example, the dominant classes for the domain Sport are: skier, cricket player, hockey player, volleyball player, swimmer, oarsman, etc., altogether 31 dominant classes. For the definition of the dominant classes, we use the WordNet sister hyponyms at a certain level (the lowest level allowing classification without specific knowledge for the domain). So far, the selected dominant classes for all thematic domains in focus are 277.

For each dominant class a parent class is selected from the WordNet noun hierarchies and this procedure is repeated consecutively up to the final class which represents a visual object. For example, classes like basketball player, acrobat, football player, etc. are hyponyms of athlete ‘a person trained to compete in sports’. Athlete in its turn is a hyponym of contestant ‘a person who participates in competitions’ which is a hyponym of person. However, the hypernym of person is organism, an abstract notion, which is not included in the ontology. As a result of this approach, thousands of annotations will be assigned to objects representing small number of classes, while the annotations with more general classes will be inherited automatically.

Attributes in the ontology are classes related with the dominant ones. The type of the dominant class and the type of attribute class determine the type of the relation between them which expresses the specificity of property attribution: has instrument, wears, uses, has part, etc. For example, the attribute classes for cricketer are cricket bat, cricket ball, cricket helmet, wicket and referee, while for climber – climbing helmet, chalk bag, claiming backpack, and so on.

Relations between dominant and attribute classes are not hierarchical. For the definition of attribute classes, we use WordNet relations such as meronymy and morpho-semantic relations between nouns. In many cases, such relations are not overtly established in WordNet and they were additionally inserted in the Ontology.

Finally, we made some evaluation tests for all selected classes with other sources providing lists with concrete objects, such as concreteness ratings (Brysbaert et al., 2018) and acquisition ratings of words (in our case of nouns) (Kuperman et al., 2012). So far, we have identified 1,037 classes grouped in ten thematic domains: Sport, Medicine, Arts, Education, Food, Transport, Clothing, Security, Indoors, Nature.

The relations used in the Ontology are relations between classes. Part of the relations and their properties are inherited form WordNet. Additional relations are included in the ontology in case they are not explicit in the WordNet structure. Each class in the Ontology is represented by a unique label, which in most cases is one of the synonyms in the corresponding WordNet synset (in case of ambiguity, a descriptive label is constructed).

Benefits of using an ontology for image labelling can be outlined as follows:

- Selection of mutually exclusive classes.
- Build-in interconnectivity of classes by means of formal relations.
- Easy extension of the proposed ontology with more concepts corresponding with visual objects.

What it more, since wordnets for many languages are linked to Princeton WordNet (Bond et al., 2016), we will provide multilingual descriptions of the images. Freely available wordnets with various lexical coverage for 17 official EU languages (Bulgarian, Croatian, Danish, Dutch, English, Finnish, French, Greek, Italian, Lithuanian, Polish, Portuguese, Romanian, Slovak, Slovene, Spanish, Swedish) and for Albanian, Icelandic, Hebrew and Serbian will be used for a multilingual representation of the selected classes.

4 Image Collection

There are many repositories that can be used for searching and downloading images. Some of the images are assigned with multiple labels or short descriptions, which is used to facilitate the automatic collection of appropriate images. For the selected classes a focused web image search is being conducted to compile a database with images — candidates for annotation. So far, more than 450,000 images were collected from different image providers, which are selected on the basis of the following criteria: the repositories should offer an API and images should be licensed with one of

13http://compling.hss.ntu.edu.sg/omw/
the following standards: Universal Public Domain Dedication (CC0 1.0); Attribution 4.0 International (CC BY 4.0) and Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)\textsuperscript{14}. Thus, we are avoiding copyrighted material, which might limit the use of our dataset only for academic purposes.

After the collection of images, we perform additional manual selection to ensure the quality of the dataset. The following criteria for selection are observed:

- The image has to contain a clearly presented object described by a given dominant class.
- The object should not (preferably) have occluded parts. If there are occluded parts of the object, they should not be essential for its recognition.
- The target object should be in its usual environment and in a position or use that is normal for its activity or purpose (for example, images in which a skier drinks a beer are not selected).
- The target object should be represented with its inherent attributes (for example, images of a man with wings are not selected).
- The target object should be represented in different positions, photographed from diverse viewpoints and angles and the object background should vary to a sufficient degree (for example, images of a chess player which slightly differ from one another are not selected).
- The instances of the target object should not represent one and the same person, animal or artefact.
- (Preferably) images with up to 10 objects are selected (the objects can belong to different classes or can be instances of one and the same class). If there are images with only one object, then it should be the dominant one.
- Images with small objects, unfocused objects in the background or images with a low quality (low resolution; blurriness caused by an out-of-focus lens, low illumination level, etc.) are not selected.
- Images which represent collages of photos, drawings or are post-processed are not selected.

The final selection of images is triple checked independently by different experts: after the automatic collection, after the automatic generation of segmentation masks and after the manual annotation: correction of the segmentation masks, new polygon outlines and selection of appropriate classes.

5 Annotation Conventions

Our aim is to provide a dataset that will support image classification, instance segmentation and object detection formats. Several open source tools for image annotation (Makes Sense\textsuperscript{15}, COCO Annotator\textsuperscript{16}, VGG Image Annotator\textsuperscript{17}, LabelMe\textsuperscript{18}, LabelImg\textsuperscript{19}, etc.) have been evaluated in order to choose the most appropriate one for our purposes. Each annotation tool is usually designed for a specific application and for a specific annotation process. For example, we experimented with the web-based image annotation tool LabelMe to create polygon annotations; with the desktop annotation tool LabelImg to create bounding boxes, etc.

To avoid converting annotations for frameworks such as YOLACT\textsuperscript{20}, DETECTRON\textsuperscript{21}, etc., which provide segmentation masks and require COCO formatted annotations, we decided to work with the COCO annotator\textsuperscript{22}. The COCO Annotator can be containerised, allows for simultaneous work on a project, and offers useful functions that facilitate image annotation: tracking object instances, labelling objects with disconnected visible parts, etc.

It is a known fact that semi-automatic annotation approaches can significantly speed up the annotation process by automatic generation of annotation proposals to support the annotators. The main idea is to reduce the human interaction with the annotation tool and to save time, while maintaining the quality of the annotations. We experimented with Mask R-CNN and YOLACT, which provides instance segmentation on datasets like COCO. Mask R-CNN (He et al., 2017) is an implementation based on Python 3, Keras and TensorFlow. The model generates bounding boxes and segmentation masks for each instance of an object in the image. YOLACT (Bolya et al., 2020) is a framework, which breaks up instance segmentation into two parallel tasks: a) generating a dictionary of non-local prototype masks over the entire image, and b) predicting a set of linear combination coefficients

\textsuperscript{14}https://search.creativecommons.org

\textsuperscript{15}https://www.makesense.ai

\textsuperscript{16}https://github.com/jsbroks/coco-annotator

\textsuperscript{17}https://www.robots.ox.ac.uk/ vgg/software/via/

\textsuperscript{18}http://labelme.csail.mit.edu/Release3.0/

\textsuperscript{19}https://github.com/tzutalin/labelImg

\textsuperscript{20}https://github.com/dbolya/yolact

\textsuperscript{21}https://github.com/facebookresearch/Detectron

\textsuperscript{22}https://github.com/jsbroks/coco-annotator
per instance. Since the number of prototype masks is independent of the number of classes (e.g., there can be more classes than prototypes), YOLACT learns a distributed representation in which each instance is segmented with a combination of prototypes that are shared across classes.

The task for the annotators is to outline polygons for individual objects in the image (either by approving or correcting the automatic segmentation or by creating new polygons) and to classify the objects against the classes from the predefined Ontology.

The annotation follows the following conventions (only the more significant ones are listed here):

- The predicted polygons are accepted or corrected (if necessary) so that they outline the object as well as possible. Every instance of the target object is provided with a polygon.
- All objects from the selected dominant class and attribute classes related with it are annotated with polygons (for example, the tennis player and the related objects racket and tennis ball; chess player and the related objects chessman, chessboard, and clock). Other objects can be also annotated if they belong to the predefined Ontology of visual objects.
- Every polygon is required to be as close to the object outline as possible. There is not much information how the overlapping objects should be annotated. The bounding boxes that embrace the estimated extent of the object are not annotated due to the ambiguity and disagreement between the annotators (Lin et al., 2014). One possible solution is to annotate only the visible parts of the objects. We accepted the following conventions: If the objects are included in each other, both objects are annotated; If two objects overlap and the boundaries of the partially occluded object are clear, then the second one is annotated with an estimation for the occluded part (for example, a car behind a road sign); In case the occluded parts cannot be determined unambiguously, they are not annotated.
- An object is not annotated if it cannot be recognised for various reasons or less than 10–20 percent of the object is visible.
- If the object can be additionally associated with a different class this is recorded within the metadata (for example, if the climber is not a man but a boy, woman or a girl).

The quality control is provided by a second annotator who validates the implementation of the conventions and discusses the quality with the annotation group weekly. If necessary, some of the images are re-annotated.

6 Conclusion and Future Work

The Multilingual Image Corpus will provide pixel-level annotations for the selected dominant classes and their parent and attribute classes in ten thematic domains, thus offering more data to train models specialised in object detection, segmentation and classification in these domains. The selected classes for annotation are organised in an Ontology of visual objects that provides options to organise annotated images in different datasets regarding the envisaged tasks.

The Multilingual Image Corpus will be released in autumn of 2021 and will provide: a) a large number of copyright-free images, b) a large number of object classes organised in an ontology, c) a large number of pixel-level annotations; and d) extended image descriptions in (at least) 20 languages based on WordNet. An important result with great significance for the development of different applications for image processing will be the open distribution of the collection.

We are currently planning some experiments with a set of state-of-the-art algorithms on each of the tasks of object detection and segmentation, in order to establish a common baseline for future work.

7 Acknowledgments

The Multilingual Image Corpus (MIC 21) project was supported by the European Language Grid project through its open call for pilot projects. The European Language Grid project has received funding from the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement no. 825627 (ELG).

References


Jia Deng, Wei Dong, Socher Richard, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. page 248–255.


Xian-Sheng Hua and Jin Li. 2015. Prajna: Towards recognizing whatever you want from images without image labeling. In AAAI International Conference on Artificial Intelligence, page 137–144.


ELERRANT: Automatic Grammatical Error Type Classification for Greek

Katerina Korre†, Marita Chatzipanagiotou†, John Pavlopoulos†♣
† Athens University of Economics and Business, Greece
♣ Stockholm University, Sweden
katkorre95,marita.xatzh,annis@aueb.gr

Abstract

In this paper, we introduce the Greek version of the automatic annotation tool ERRANT (Bryant et al., 2017), which we named ELERRANT. ERRANT functions as a rule-based error type classifier and was used as the main evaluation tool of the systems participating in the BEA-2019 (Bryant et al., 2019) shared task. Here, we discuss grammatical and morphological differences between English and Greek and how these differences affected the development of ELERRANT. We also introduce the first Greek Native Corpus (GNC) and the Greek WikiEdits Corpus (GWE), two new evaluation datasets with errors from native Greek learners and Wikipedia Talk Pages edits respectively. These two datasets are used for the evaluation of ELERRANT. This paper is a sole fragment of a bigger picture which illustrates the attempt to solve the problem of low-resource languages in NLP, in our case Greek.

1 Introduction

Grammatical Error Correction (GEC) is the task of automatically correcting language mistakes in written texts. These mistakes can vary from grammatical mistakes to punctuation, spelling and morphology of a word. The development of a GEC system usually involves the transformation of an erroneous sentence into its correct version, while also keeping the initial meaning intact. Developing those systems requires error annotated data, which can be either learner data or artificial. High-resource languages, such as English, present a variety of learner data that cover a relatively wide spectrum of language proficiency levels, native language and topics. Some notable examples are the Cambridge English Write & Improve corpus (Yannakoudakis et al., 2018), the LOCNESS corpus (Granger, 1998), and the NUCLE corpus (Dahlmeier et al., 2013). Low-resource languages, on the other hand, are characterized by a scarcity of such corpora, as well as of other GEC resources and Natural Language Processing (NLP) tools. That is also the case of Greek.

Although, Greek is only spoken by approx. 13.5 million people (native), the fact that Greece has a high immigrant population underlines the need for learning Greek as a Second Language (GSL). Therefore, and as technology is being integrated in education (Meurers, 2012; Forcier, 2016), the need for more GEC and NLP tools for Greek becomes evident.

In this paper, we present ELERRANT, an automatic annotation tool, which is based on ERRANT (Bryant et al., 2017). ERRANT produces an annotation mainly consisting of the error location, the error type and the correction of the error, by using an original erroneous sentence along with its correction as input. ERRANT is the first toolkit that not only annotates texts but also provides automatic error typing, offering detailed feedback to Second Language (L2) learners and useful information for language analysis (Bryant et al., 2017). Most importantly, the annotator’s workload is relieved and all learner corpora, regardless of size, level and other factors can be annotated in a standardized manner. We believe that its easy application and versatility can encourage the generation of more error annotated datasets in the Greek language, thus tackling the scarcity of resources.

For the evaluation of ELERRANT we developed two datasets: the Greek Native Corpus (GNC) and the Greek Wiki Edits (GWE). GNC comprises native Greek student essays, while GWE comprises sentences extracted from WikiConv (Hua et al.,

1https://en.wikipedia.org/wiki/Greek_Language
2https://eacea.ec.europa.eu/national-policies/eurydice/content/population-demographic-situation-languages-and-religions-33_en
By evaluating ELERRANT on the latter corpus, we also show that the tool has the potential to detect edits and alterations on Wikipedia Talk Pages that are due to grammatical error correction, which can then be automatically white-flagged from being moderated for misinformation (e.g., for words introducing bias). It also paves the way for further analysis of such edits. Both datasets are shared for public use.3

The rest of the paper is structured as follows: First, we discuss related work on Error Annotated Data and the original ERRANT. Then, we describe the development of ELERRANT. Section 4 introduces the two new datasets, demonstrates our method of evaluation and presents the findings. Section 5 is concerned with the use and implications of ELERRANT. Finally, we conclude by discussing limitations and future work.

2 Related Work

Error Annotated Data - What for? Error annotated data can be useful in multiple domains ranging from real-life teaching and educational research to NLP tasks, especially in GEC. More specifically, recent advances in GEC often require large amounts of annotated data both for development and evaluation of any given systems. Mita et al. (2019) underline the need for cross-corpora evaluation when it comes to GEC systems, given that the task difficulty depends on factors such as proficiency level and essay topic. Consequently, there is a demand for standardized error-annotated corpora, while also reducing the annotator’s workload (Bryant et al., 2017). In addition, error annotated corpora can play a major role in error analysis, which has slowly started to step into CALL (Computer Assisted Language Learning). Until very recently, the staple technique to NLG (Natural Language Generation) for language learning purposes, was to train models on large bodies of correct English (Lee and Seneff, 2008). Although this technique has proven to be effective, a more recent one seems to take into account more parameters when it comes to non-native speakers. This new technique involves relying on two kinds of corpora: a source corpus from non-native texts, and a target corpus, which, in reality, is a corrected version of the source corpus. Meurers (2012) summarizes the benefits on the analysis of learner corpora claiming that the annotation of learner corpora can point out learner language properties thus supporting the aim of improving our understanding of Second Language Acquisition (SLA) and developing instructional methods and materials for SLA purposes.

ERRANT Bryant et al. (2017) attempt to solve the issue of corpora standardization by presenting ERRANT, an automatic annotation tool which serves as both annotator and system-output scorer. ERRANT only needs an erroneous sentence along with its correction to produce an annotation essentially consisting of the location of the errors, the error type, and the correction. ERRANT has paved the way for a new annotation framework, and has therefore been used in the most recent shared task, the BEA-2019 (Bryant et al., 2019), both for annotating the datasets used for the task and for evaluating the system output of the participants per error. The convenience and versatility of ERRANT has led to the adaptation of the tool in more languages, such as German (Boyd, 2018), Spanish (Davidson et al., 2020), Czech (Náplava and Straka, 2019), and Romanian (Cotet et al., 2020). In this paper, we present our Greek version.

Task difficulty Inter-annotator agreement, although a staple in computational linguistics procedures when it comes to evaluation, has been quite controversial regarding GEC. Traditionally, corpora for GEC purposes would be annotated by solely one native annotator providing one gold standard annotation, a practice which automatically renders the research highly biased and even uninformative (Bryant and Ng, 2015; Tetreault and Chodorow, 2008). The obvious solution to the problem would be to recruit multiple annotators and estimate the degree to which they agree on the correction of an error. Yet, this method also proves insufficient, since annotators usually agree up to 70%, a percentage inadequate for system evaluation. The same holds for intra-annotator agreement, where the same annotator does not always agree with themselves (κ scores of about 60%) (Bryant and Ng, 2015).

3 The datasets are shared with CCO licence on: https://github.com/katkorre/elerrant

3https://sourceforge.net/projects/grspell/files/hunspell-gr

3The datasets are shared with CCO licence on: https://github.com/katkorre/elerrant

3https://sourceforge.net/projects/grspell/files/hunspell-gr
<table>
<thead>
<tr>
<th>Error Type</th>
<th>Meaning</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD:FORM*</td>
<td>Adjective Form</td>
<td>Errors concerning the form of an adjective</td>
<td>καλός → καλοτέρος</td>
</tr>
<tr>
<td>NOUN:FORM</td>
<td>Noun Form</td>
<td>Errors concerning the number, the case or the suffix of a noun.</td>
<td>του νους → τον νου</td>
</tr>
<tr>
<td>PRON:FORM</td>
<td>Pronoun Form</td>
<td>Errors concerning the number, the case or the suffix of a pronoun.</td>
<td>καθηκοντα → καθηκοντας</td>
</tr>
<tr>
<td>VERB:FORM</td>
<td>Verb Form</td>
<td>Errors concerning the disposition, the voice, the inflection, the tense, the number or the person of a verb.</td>
<td>(εσείς) πηγαίνεται → (εσείς) πηγαίνετε</td>
</tr>
<tr>
<td>CONJ</td>
<td>Conjunction</td>
<td>Errors concerning conjunctions.</td>
<td>καν → αλλα</td>
</tr>
<tr>
<td>PREP</td>
<td>Preposition</td>
<td>Errors concerning prepositions.</td>
<td>από → σε</td>
</tr>
<tr>
<td>DET*</td>
<td>Determiner</td>
<td>Errors concerning articles or determiners.</td>
<td>το → του</td>
</tr>
<tr>
<td>SPELL</td>
<td>Spelling</td>
<td>Spelling errors.</td>
<td>συγγέρα → συγγέρα</td>
</tr>
<tr>
<td>FN</td>
<td>Final -ν/μ</td>
<td>Final -ν/μ addition or removal.</td>
<td>θην → τη/μη → μη</td>
</tr>
<tr>
<td>PUNCT</td>
<td>Punctuation</td>
<td>Errors concerning the punctuation.</td>
<td>. → ;</td>
</tr>
<tr>
<td>OTHER</td>
<td>Other Errors</td>
<td>An error that does not fit into any other category but can still be corrected.</td>
<td>χάμα → για χανέα</td>
</tr>
<tr>
<td>ACC</td>
<td>Accentuation</td>
<td>Accentuation addition or removal.</td>
<td>καθήκοντα → καθήκοντα</td>
</tr>
<tr>
<td>UNK</td>
<td>Unknown error type</td>
<td>An error that can be detected but not corrected.</td>
<td>usually long error spans</td>
</tr>
<tr>
<td>WO</td>
<td>Words Order</td>
<td>Error in words order.</td>
<td>οταν φέρεις αρφες → οταν έφερες φέρνη</td>
</tr>
<tr>
<td>ORTH*</td>
<td>Orthography</td>
<td>Spacing Errors</td>
<td>γιασένα → για σένα</td>
</tr>
<tr>
<td>PART:FORM</td>
<td>Participle Form</td>
<td>Errors concerning the number, the case or the person of a participle.</td>
<td>(πηγή) τρέχοντας → (πηγή) τρέχοντα</td>
</tr>
<tr>
<td>VERB:SA</td>
<td>Subject Verb Agreement</td>
<td>The subject and the verb to be in person agreement.</td>
<td>(έγώ θα) φύγει → (έγώ θα) φύγω</td>
</tr>
</tbody>
</table>

Table 1: ELERRANT and human error type annotation guide. The error types with the asterisk (*) do not exist in the human annotation scheme while the two last error types in the table do not exist in the ELERRANT annotation scheme.

SpaCy\(^3\) as the main POS tagger. Due to morphological differences between the two languages (English and Greek), we removed some error categories that exist in the original ERRANT, while adding some new ones. Due to the fact that Greek is a highly inflectional language and most POS have some sort of inflection, we “merged” some error types in order to include as much information about the error as possible. This decision can be regarded as a compromise, since many errors might have more than one overlapping error types (e.g., τον γάτα → τον γάτιων, wrong case and number), therefore by merging the sub-types into the FORM type we preserve the ambiguity and multifaceness of the error.

The main alterations are the following: We added the error type AD:FORM (Adverb Form), to convey errors that mainly concern the comparative and superlative degree of the adverb. The CONTR (Contraction) category has been removed temporarily due to the fact that contractions in Greek can happen in any word starting or ending with a vowel under certain conditions. We are currently developing a dictionary that assembles the most frequent cases of contractions in Greek and we plan to integrate it in future versions. NOUN:INFL (Noun Inflection), NOUN:POSS (Noun Possessive) and NOUN:NUM (Noun Number) are all captured in NOUN:FORM (Noun Form). PART (particles) were also dropped as in the Greek language they have a different function, mainly in tense construction. PRON:FORM (Pronoun Form) was also added, since pronouns are also inflectional. From the verb categories, only VERB, VERB:FORM, and VERB:TENSE have been preserved.

**Two new categories** We added two more categories from scratch: ACC (Accent) and FN (Final -ν/μ). The accent in Greek is signified with a stress mark (´) rather than just the intonation of the word when speaking. The maintenance or omission of the final nu in some Greek words (articles, pronouns or particles), due to its frequency as an error even by native speakers, is considered a different error type and not just a spelling error. For the two aforementioned error types, we made two

---

\(^3\)https://spacy.io/models/el
new corresponding functions and added them in the ELERRANT classifier (see Algorithms 1 and 2). Table 1 demonstrates the final error categories in ELERRANT.

Table 1 demonstrates the final error categories in ELERRANT. If a sentence contains at least one error, the corresponding error categories are documented in the table. This paper, 227 sentences have been collected and annotated. The GNC comprises essays written by High School students.

Algorithm 1: Accent error detection

<table>
<thead>
<tr>
<th>Data:</th>
<th>chars \textsuperscript{orig}, chars \textsuperscript{corr}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result:</td>
<td>label ∈ {R:ACC, M:ACC, U:ACC}</td>
</tr>
<tr>
<td>accents = {α, ι, ε, ο, η, ῑ};</td>
<td></td>
</tr>
<tr>
<td>accents\textsuperscript{orig} = chars\textsuperscript{orig} \cap accents;</td>
<td></td>
</tr>
<tr>
<td>accents\textsuperscript{corr} = chars\textsuperscript{corr} \cap accents;</td>
<td></td>
</tr>
<tr>
<td>if accents\textsuperscript{orig} ≠ {} then</td>
<td></td>
</tr>
<tr>
<td>// Only the original word has an accent</td>
<td></td>
</tr>
<tr>
<td>if accents\textsuperscript{corr} ≠ {} then</td>
<td></td>
</tr>
<tr>
<td>return M:ACC;</td>
<td></td>
</tr>
<tr>
<td>else</td>
<td></td>
</tr>
<tr>
<td>// Both words have accents, so compare the number of accents between them</td>
<td></td>
</tr>
<tr>
<td>if len(accents\textsuperscript{orig}) &gt; 1 then</td>
<td></td>
</tr>
<tr>
<td>// Redundant accent in the original</td>
<td></td>
</tr>
<tr>
<td>return U:ACC;</td>
<td></td>
</tr>
<tr>
<td>else if len(accents\textsuperscript{orig}) = 1 then</td>
<td></td>
</tr>
<tr>
<td>// Missing accent in the original</td>
<td></td>
</tr>
<tr>
<td>return M:ACC;</td>
<td></td>
</tr>
<tr>
<td>else if accents\textsuperscript{orig} ≠ accents\textsuperscript{corr} then</td>
<td></td>
</tr>
<tr>
<td>// Same number of accents, yet different</td>
<td></td>
</tr>
<tr>
<td>return R:ACC;</td>
<td></td>
</tr>
</tbody>
</table>

Algorithm 2: Final \(\nu\) error detection

<table>
<thead>
<tr>
<th>Data:</th>
<th>chars \textsuperscript{orig}, chars \textsuperscript{corr}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result:</td>
<td>label ∈ {M:FN, U:FN}</td>
</tr>
<tr>
<td>// Original token: the corrected + (\nu)</td>
<td></td>
</tr>
<tr>
<td>if chars\textsuperscript{orig} = chars\textsuperscript{corr}[; -1] then</td>
<td></td>
</tr>
<tr>
<td>if chars\textsuperscript{corr}[−1] = “(\nu)” then</td>
<td></td>
</tr>
<tr>
<td>// The original is missing the final (\nu)</td>
<td></td>
</tr>
<tr>
<td>return M:FN;</td>
<td></td>
</tr>
<tr>
<td>else</td>
<td></td>
</tr>
<tr>
<td>// The other way, unnecessary final (\nu)</td>
<td></td>
</tr>
<tr>
<td>if chars\textsuperscript{corr}[−1] = “(\nu)” then</td>
<td></td>
</tr>
<tr>
<td>return U:FN;</td>
<td></td>
</tr>
</tbody>
</table>

4 Empirical Evaluation

We consider this work an opportunity to introduce two novel datasets: Greek WikiEdits (GWE), and the Greek Native Corpus (GNC), which we use as gold standards in our ELERRANT evaluation. This section first describes our two datasets, their annotation and development process, and presents some statistics and inter-annotator agreement results. Then, the evaluation is discussed and the experimental results are reported, evaluating ELERRANT on both datasets.

4.1 Datasets

GWE The first corpus we evaluated ELERRANT on is based on WikiConv (Hua et al., 2018). WikiConv is a multilingual corpus that encompasses the history of conversations on Wikipedia Talk Pages, including comment deletion, modification and restoration. The authors of the respective article kindly provided us with the Greek part of the corpus, which comprises 194,499 Talk Pages. We processed the provided pages so that only sentences with edits remained. Despite the fact that edits do not necessarily regard grammatical errors (e.g., they could be about a corrected date), their employment by Grammatical Error Correction models leads to improvements (Lichtarge et al., 2019). To the best of the authors’ knowledge, this is the first dataset with edits of Wikipedia Talk Pages comprising human annotations for grammatical errors. Henceforth, we will refer to this Greek WikiEdits dataset as GWE.

GNC The existing publicly available corpora compilations, that attempt to contribute to the scarcity of Greek resources for NLP, are the Greek Learner Corpus (GLC) (Tantos and Papadopoulou, 2018), and the Electronic Learner Corpus of L2 Greek (Tzimokas, 2010). Both datasets consist of data generated by learners of Greek as a Second Language (GSL). Despite their usefulness, we observe that none of these datasets includes corrections in their annotations. This lack of corrections was intentional, in order to reflect the “error ambiguity”, rather than choosing between several corrections, given that an error can be corrected in multiple ways (Tantos and Papadopoulou, 2018). This lack of corrections, however, also means that ELERRANT is effectively inapplicable on them. This gap motivated us to develop another corpus to evaluate ELERRANT on, the first Native Greek Corpus (GNC), designed in the aim of being compatible to ELERRANT and of use to automatic grammatical error correction systems.

The compilation of GNC is currently in progress, but at the time of writing this paper, 227 sentences have been collected and annotated. The GNC comprises essays written by High School students,
whose native language is Greek. The hand-written essays were split into sentences and manually digitized (no OCR was used). Each sentence may contain none, one or more grammatical errors.

4.2 Annotation Schema

The GWE corpus comprised edits which we considered the “corrections”, in order to be able to apply ELLERRANT. GNC, however, does not comprise any suggested corrections and thus, ELLERRANT is not applicable. Hence, the suggested corrections have to be provided by the annotators. Due to this significant difference between the two datasets, the annotation process differs in the two cases, and in particular concerning the GNC it becomes slightly more complicated. Annotation for both datasets was based on the rule-based error type framework of the original ELLERRANT. Two main code categories were created. The first category (‘Error Description’), consists of the three prefix operation codes \([U(\text{necessary})/R(\text{replacement})/M(\text{issing})]\) which indicate what needs to happen to each erroneous item in order to be corrected, i.e., whether it should be removed from the sentence, whether it should be modified or replaced, or whether an item is missing. The second code category (‘Error Type’) comprises 16 codes created based on the 25 codes presented in (Bryant et al., 2017) (see Table 1), which form a simplified classification system of the errors that may occur in the written Greek language. These codes indicate what kind of error we encounter in each sentence and (in most cases) what part of speech the erroneous element that needs to be removed, modified or added is.

GWE Schema The annotation on the GWE was carried out in two steps. First, the two parallel sentences (original and changed) were input in ELLERRANT, which helped us extract the tokens of the edits. Then, the annotators classified the edits according to the ‘Error Description’ and ‘Error Type’ fields from the annotation schema described above.

GNC Schema The annotation process for the GNC was based on a schema containing four annotation fields. The first field is intended for two mutually exclusive values \([c(\text{orrect})/e(\text{rroneous})]\) in order to mark the absence or presence of an error in the sentence. The second is for the annotator to correct the existing error and rewrite the whole sentence providing the corrected string. The two remaining fields are respectively for the ‘Error Description’ and ‘Error Type’ codes that were used also at the annotation of GWE.

As can also be seen in Table 2, when a sentence contains multiple errors, such as in (a) below, then copies of that sentence are inserted in the corpus (i.e., the second and third), leading to as many records with the same sentence as the errors it comprises. In each entry, the annotator should maintain and correct only one error, while the other errors must be recorded in advance (both in the field of the original and the corrected text) as correct. Respectively, the same should happen if two or more errors are detected in a single word, such as the example of sentence (b) below:

(a) **Erroneous sentence containing more than one errors**

Αρχικά, από την μεριδατοποίηση και την ευκολοποίηση τους, προκαλεί σεις και προκαλούνται δυσαναλογικά αποτελέσματα. Αποτυχάνουν αποτελώντας, έτσι που σεις και προκαλούνται και προκαλούνται σεις και προκαλούνται.

(Αρχικά)

(b) **Erroneous word containing more than one errors**

Εμείς επιχηρούμαι.

"We attempt."

Annotation process We recruited two Greek philology graduates and provided them with 327 sentences, 227 from GNC and the remaining from GWE. In GWE, where each page comprised a single edit, we performed sentence segmentation (based on full stop) and only considered the sentence comprising the edit. We asked the annotators to follow the annotation schemas (see Sec. 4.2), in order to classify (and detect and correct in GNC) all possible grammatical errors.

4.3 Corpus Statistics

Table 3 presents the statistics of the two annotated corpora, by considering the detected errors of the two annotators on (micro) average. Sentences were longer in GWE, compared to GNC. More errors were detected on (micro) average in GNC, which is
Evidently, from the human-annotated error types and those generated by ELERRANT, the process was the same for GNC and GWE.

Inter-Annotator Agreement In GWE, Cohen’s Kappa for the human-annotated edits was 70.02%. Out of the sixteen available codes, only 10 were used by Annotator I and 13 by Annotator II. The Pearson’s correlation between the error type frequencies of the two annotators was 99.62%, which indicates that the two frequency distributions coincide to a large extent. Given that the annotators had to annotate the edits with annotation guidelines fitting for error types, they opted for more abstract categories such as OTHER, which was the most frequently annotated type. In GNC, Cohen’s Kappa between the two annotators regarding the error type was even higher than GWE, reaching 84.65%. Out of the sixteen available codes, in the annotated texts we encountered the fourteen (UNK and WO tags were not used at all). The distribution of the frequency of occurrence of these fourteen (14) error types also coincides to a large extent between the two annotators, which is reflected in the high error type frequency correlation between the two annotators (99.20%). Any disagreement is mainly due to additional errors and not to the incorrect or different rendering of the codes.

In other words, inter-annotator agreement in GWE falls within the threshold of 70% (see Section 2), while in the GNC this threshold is exceeded. In such cases, where agreement is not optimum but not extremely disheartening, tools such as ERRANT can be used as moderating tools by pinpointing any great discrepancies, as a third annotator would.

4.4 Experimental Results

Since GNC was annotated by two annotators (and therefore there might be differences in the corrections), we ran ELERRANT on both annotators’ corrections separately and compared the output error types against the error types provided by each annotator, which served as our gold standards. For GWE, first, the texts were input into ELERRANT and then the edits were assigned an error type by the annotators. Then, we calculated the accuracy, precision, recall and F1-score separately for each comparison. The three latter metrics were calculated with macro, micro and weighted averages to also be able to see whether ELERRANT performs better at more or less frequent error types. We also calculated and compared the frequencies of the human-annotated error types and those generated by ELERRANT. The process was the same for GNC and GWE.

<table>
<thead>
<tr>
<th>Label</th>
<th>Original Text</th>
<th>Corrected Text</th>
<th>Error Description</th>
<th>Error Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>Αρχικά, από την μεριά των μεγάλων καλό θα ήταν να κατανόησουν πως ο υπερπροστατευτισμός αναστέλλει την υπευθυνότητα των νέων και προκαλεί συνήθως έντονες αντίδρασεις, αποτυγχάνοντας έτσι την προστασία τους και προκαλούσε μάλλον αντίθετη αποτύχηση.</td>
<td>Αρχικά, από την μεριά των μεγάλων καλό θα ήταν να κατανόησουν πως ο υπερπροστατευτισμός αναστέλλει την υπευθυνότητα των νέων και προκαλεί συνήθως έντονες αντίδρασεις, αποτυγχάνοντας έτσι την προστασία τους και προκαλούσε το μάλλον αντίθετη αποτύχηση.</td>
<td>R</td>
<td>FN</td>
</tr>
<tr>
<td>e</td>
<td>Αρχικά, από την μεριά των μεγάλων καλό θα ήταν να κατανόησουν πως ο υπερπροστατευτισμός αναστέλλει την υπευθυνότητα των νέων και προκαλεί συνήθως έντονες αντίδρασεις, αποτυγχάνοντας έτσι την προστασία τους και προκαλούσε μάλλον αντίθετη αποτύχηση.</td>
<td>Αρχικά, από την μεριά των μεγάλων καλό θα ήταν να κατανόησουν πως ο υπερπροστατευτισμός αναστέλλει την υπευθυνότητα των νέων και προκαλεί συνήθως έντονες αντίδρασεις, αποτυγχάνοντας έτσι την προστασία τους και προκαλούσε το μάλλον αντίθετη αποτύχηση.</td>
<td>R</td>
<td>SPELL</td>
</tr>
<tr>
<td>e</td>
<td>Αρχικά, από την μεριά των μεγάλων καλό θα ήταν να κατανόησουν πως ο υπερπροστατευτισμός αναστέλλει την υπευθυνότητα των νέων και προκαλεί συνήθως έντονες αντίδρασεις, αποτυγχάνοντας έτσι την προστασία τους και προκαλούσε το μάλλον αντίθετη αποτύχηση.</td>
<td>Αρχικά, από την μεριά των μεγάλων καλό θα ήταν να κατανόησουν πως ο υπερπροστατευτισμός αναστέλλει την υπευθυνότητα των νέων και προκαλεί συνήθως έντονες αντίδρασεις, αποτυγχάνοντας έτσι την προστασία τους και προκαλούσε το μάλλον αντίθετη αποτύχηση.</td>
<td>R</td>
<td>PART-OF-NUM</td>
</tr>
</tbody>
</table>

Table 2: Annotation sample: three entries of the same sentence annotating each time a different error.

<table>
<thead>
<tr>
<th>ANNOTATED SENTENCES (#)</th>
<th>GWE</th>
<th>GNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR ANNOTATIONS (#)</td>
<td>100</td>
<td>180</td>
</tr>
<tr>
<td>TOKENS PER ANN. SENTENCE (#)</td>
<td>46,43</td>
<td>21,2</td>
</tr>
<tr>
<td>COHEN’S KAPPA (%)</td>
<td>70.02</td>
<td>84.65</td>
</tr>
</tbody>
</table>

Table 3: Overview of GWE (Talk Pages) and the GNC (essays). The respective count is shown per row.
It must also be noted that to evaluate the performance of ELERRANT against the gold standards of both datasets, we had to mitigate the differences between the ELERRANT annotation schema and the human annotator schema (see Table 1). The necessary modifications included changing all \texttt{VERB:*} error types to \texttt{VERB:FORM}, \texttt{ARTORDET} to \texttt{DET}, \texttt{PART:FORM} to \texttt{ADJ:FORM}, and temporarily change \texttt{ADJ:FORM} of ELERRANT to \texttt{AD:FORM}.

**Evaluation on WikiEdits (GWE)** For the evaluation of ELERRANT on GWE, as already discussed, we considered the annotations of each annotator as our ground truth. The results are demonstrated on the right of Table 4. All metrics are low with the accuracy reaching 31% and 27%, precision never exceeding 44.47%, recall 40.03% and F1-score 35.75%. This low performance is explained in part by the inability of ELERRANT to detect specific types (see Fig. 1). Additionally, as mentioned in Section 4.1, the edits in Greek Wikipedia are not necessarily grammatical errors. For instance, edits can be ‘vandalisms’, attempting to alter the content and context of the respective Wikipedia article.

**Evaluation on Learner Data (GNC)** GNC results are presented on the left of Table 4. Scores are considerably high for micro and weighted averaging. Error type classification is a multi-class problem and each instance of error type might be encountered in different frequencies. This is also apparent in Figure 2, where we can see that spelling, accent, and final nu errors have the highest frequencies both in terms of ELERRANT annotation and according to human annotators. In both annotator cases, micro scores are higher than macro scores indicating that ELERRANT performs better when it comes to classifying an error type that occurs frequently, while it tends to misclassify less frequent error types. The accuracy scores, 83.91% and 77.30%, by Annotator I and II, respectively, show that ELERRANT can correctly classify the error type approx. eight out of ten times, assigning the most appropriate error type possible, if we consider the annotation as the gold standard, and thus the best possible error type classification. However, and as we can see from the results, there are discrepancies between the two annotators, hence the different accuracy scores (see Section 4.3).

Taking into consideration that ground truth in GEC annotation cannot be as established as in other NLP tasks, due to the fact that one erroneous sentence can have multiple corrections (Napoles et al., 2015; Bryant and Ng, 2015), we also looked at the ELERRANT output manually to pin down where it is lacking exactly. We noticed two major issues: First, Greek SpaCy (which is the core of ELERRANT) does not perform as well as in the English version, assigning wrong POS tags or dependencies. Secondly, there are cases in Greek where a single word can contain two or more errors at the same time, and cannot be solved with the ‘FORM’ category. These cases are usually a combination of accent and spelling mistakes, spelling mistakes and morphology, accent and morphology, etc. In this case, ELERRANT assigns only one error type disregarding the rest. We hope to solve these issues in future versions.

### 5 Discussion

Our experimental results showed that ELERRANT performs better when it comes to actual error type classification (GNC) than edit classification (GWE). In terms of scores, the ELERRANT classifier works adequately (83% accuracy), especially when compared to the evaluation of the original ERRANT, for which a manual evaluation rated 86% of the output error types as “GOOD”. There is still, however, room for improvement. As mentioned in Section 4, the most important issue is Greek SpaCy which also hinders the development of a more detailed ELERRANT, i.e., with more error types such as Noun-Gender Agreement, Case, etc.

As far as the GWE is concerned, a detection and edit classification by ELERRANT is possible, provided that both the edit and the proper form of the text exist. Moreover, the addition of further edit categories is necessary because the current version of ELERRANT is based on error type classification and not edit classification. Categories such as SYNONYM and NAME might give a better insight into how the edits affect the text.

Table 5 illustrates the potential, as well as the problem of applying ELERRANT to GWE. The first edit is a name replacement, which does not entail a grammatical error, yet it does affect the meaning of the sentence. ELERRANT incorrectly classified the edit as a spelling error, while the two annotators placed it in the more general category OTHER. If both ELERRANT and the human-annotator scheme provided a category such as \texttt{R:NAME}, the performance of the system would have been better and the human annotation would have been more accurate.
Table 4: ELERRANT evaluation using Precision, Recall, F1, Accuracy in classifying GWE and GNC error types.

<table>
<thead>
<tr>
<th></th>
<th>GNC</th>
<th>GWE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ANNOTATOR I</td>
<td>ANNOTATOR II</td>
</tr>
<tr>
<td></td>
<td>Macro</td>
<td>Micro</td>
</tr>
<tr>
<td>Precision</td>
<td>58.15</td>
<td>86.90</td>
</tr>
<tr>
<td>Recall</td>
<td>60.92</td>
<td>83.91</td>
</tr>
<tr>
<td>F1</td>
<td>58.54</td>
<td>85.38</td>
</tr>
<tr>
<td>Accuracy</td>
<td>-</td>
<td>83.91</td>
</tr>
</tbody>
</table>

Figure 1: Frequencies of error types on GWE inferred by ELERRANT (A1 ELERRANT, A2 ELERRANT) against those of the two annotators (A1 Gold, A2 Gold).

Figure 2: Frequencies of error types on GNC inferred by ELERRANT (A1 ELERRANT, A2 ELERRANT) against those of the two annotators (A1 Gold, A2 Gold).

Table 5: Example annotated sentences from GWE.

Original/Edit | ELERRANT | Annotator I | Annotator II |
---------------|----------|-------------|--------------|
Σομαλίας / Γροιλανδίας | R:SPELL | R:OTHER | R:OTHER |
the (definite feminine article) or harpafr/pe/ərpafr daily | R:SPELL | R:SPELL | R:NOUN:FORM |

leading to a more accurate and descriptive ground truth. When the edit is a grammatical error ELERRANT performs better, as we can see in the second example. Finally, there is the case where ELERRANT is right and the human-annotator is wrong, which also indicates that an automatic annotation tool such as ERRANT and ELERRANT can provide more accurate, less biased annotation.

6 Conclusion

This paper presented ELERRANT, the Greek version of the automatic grammatical error type annotation tool ERRANT. With this work we also introduced two new datasets: the GNC and GWE, which can be used for GEC and edit classification purposes. Both our datasets are released for public use. In GNC, our findings showed that ELERRANT achieves an accuracy of 77.30%-83.91%, confirming that it can be an effective tool, reducing the scarcity problem of low-resource languages, such as Greek. In GWE, the overall performance was much lower, mainly because Wikipedia edits are not necessarily due to GEC. However, despite this low performance, we observe that ELERRANT can still be helpful to Wikipedia moderators who can use it to shortlist edits that are likely due to GEC and thus white-list them. Furthermore, ELERRANT could be updated to capture more error types that are common in GWE, which we will consider in future work, along with the expansion of our datasets.
7 Ethical Considerations

All texts for the compilation of the GNC dataset were obtained with the consent of the original authors. In case of underage authors, adult parents or guardians gave their consent. Authors were thoroughly informed about the purpose of the study, and became completely aware that the produced texts would be anonymously published.

Acknowledgments

We would like to thank Jeffrey Sorensen (Jigsaw) for contributing to this research by sharing with us the Greek part of the WikiConv corpus, which led to the creation of GWE. We would also like to thank Eleutheria Stroumbouli and Maria Fasoi (Athens University of Economics and Business) for correcting and annotating the GNC and GWE datasets.

References


Christopher Bryant and Hwee Tou Ng. 2015. How far are we from fully automatic high quality grammatical error correction? In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 697–707, Beijing, China. Association for Computational Linguistics.


Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint


Neural Machine Translation for Sinhala-English Code-Mixed Text

Archchana Kugathasan  
Department Of Computational Mathematics  
University Of Moratuwa  
Katubedda  
arachchanakugathasan@gmail.com

Sagara Sumathipala  
Department Of Computational Mathematics  
University Of Moratuwa  
Katubedda  
sagaras@uom.lk

Abstract

Code-mixing has become a moving method of communication among multilingual speakers. Most of the social media content of the multilingual societies are written in code-mixed text. However, most of the current translation systems neglect to convert code-mixed texts to a standard language. Most of the user written code-mixed content in social media remains unprocessed due to the unavailability of linguistic resource such as parallel corpus. This paper proposes a Neural Machine Translation (NMT) model to translate the Sinhala-English code-mixed text to the Sinhala language. Due to the limited resources available for Sinhala-English code-mixed (SECM) text, a parallel corpus is created with SECM sentences and Sinhala sentences. Srilankan social media sites contain SECM texts more frequently than the standard languages. The model proposed for code-mixed text translation in this study is a combination of Encoder-Decoder framework with LSTM units and Teachers Forcing Algorithm. The translated sentences from the model are evaluated using BLEU (Bilingual Evaluation Understudy) metric. Our model achieved a remarkable BLEU score for the translation.

1 Introduction

Before 1990, translation was considered a difficult task due to many reasons such as ambiguity, translation mismatch, co-reference, translation divergence and development of language over time (Sreeleekha et al., 2016) but Machine Translation (MT) since 1990 has been a vast and successful research area in natural language processing. Machine translation has been given importance in the research field because it is used to translate texts for military authorities to track enemies, foreign business collaborations, marketing, etc. (Kalchbrenner and Blunsom, 2013).

Expressing the thoughts of the personal interests, daily life etc., of a person in social media networks has become a trending activity among people. Texts extracted from social media lead to measure the social dynamics of several societies (Arguello et al., 2008). Content-based search engines, personalized advertisements, recommendation systems, etc. use the user-generated content from social media to increase their value and to provide more accurate results to the users (Sippel and Brodt, 2008). Processing a content in standard language (without code-mixing) is considered as an easy task, where the text with code-mixing has been considered as a road block to extract the needed information. Code-mixing is considered as an invention of bilingualism and multilingualism. The capability of speaking in two languages, is called bilingualism and more than two languages is called multilingualism. Most of the Srilankans are bilingual. The user-generated content such as posts, comments, reviews etc., in Srilankan social media are mostly in SECM text. The main focus of this study is to translate the SECM text to Sinhala language.

To understand the necessity of processing SECM text, a survey study was conducted among 82 native Sinhala speaking citizens as a part of our research study to collect information about the usage of SECM text in Sri Lanka. Figure 1 shows the results of a few essential questions from the survey. The majority of the people have stated that they use SECM text in social media rather than their native language.
In SECM texts there are several problems identified related to morphology, syntax and the semantic structure of the text. The challenges found in SECM texts are,

- Inconsistency in the transliteration
- Spelling mistakes
- Code-switching
- Words combined with suffixes of another language
- Improper usage of discourse marker
- Unnecessary numerical characters combined with words

For example, in the SECM sentence shown in Figure 2, the words ‘lassana’, ‘ekak’, ‘ekka’ and ‘gatha’ are transliterated (Words from one language written with the alphabet of another language) Sinhala words. The word ‘atmosphere’ and ‘so’ are English words. The sentence starts with Sinhala transliterated word, switches to English and again switches back to Sinhala transliterated format. The language of words is switched from one to another in a single sentence, called as code-switching. The word ‘so’ is a discourse marker in English, which is used for joining two sentences which has Sinhala base. The ‘friendla’ is a SECM word, where English singular noun ‘friend’ is combined with Sinhala transliterated suffix ‘la’ to make it look like the plural word ‘Friends’. The word ‘4to’ represent the English word ‘Photo’ with spelling mistakes and unnecessary numerical character. The numerical character combined with the word presents the phonetic sound of the word ‘four’ and ‘to’. Together it is understood as the word ‘photo’. The research study of Kugathasan and Sumathipala (2020) clearly explains the challenges in Sinhala-English code-mixed texts.

This paper is divided into six sections. Section 2 elaborates on the works related to Machine Translation. Section 3 describes the SECM texts and corpus creation. The Section 4 explains the methodology, which explains about the model implementation and prediction. Section 5 describes the experimental setting of the model and the result gained. As the final section the paper discusses the conclusion of the research study.

Figure 1: Survey results on Sinhala-English code-mixed text usage among Srilankans
2 Related Work

The high demand for Machine translation is increased due to many reasons such as business in overseas, tracking down the information of another language for military services, high usage of social media, etc.

Machine translation was initiated by Warren Weaver in 1955. The research combined Statistical Machine Translation (SMT) with Claude Shannon’s information theory (Weaver, 1949/1955). SMT models output the degree of similarity between the source and target sentences (Carrera et al., 2009). The structure of a sentence, feature engineering and design are considered as valuable factors in SMT. Also SMT approach is noted as ‘not suitable’ for generalized sentence pair, sentences with hidden details (Kalchbrenner and Blunsom, 2013) and language pairs with different word orders (Masoud et al., 2019). Chiang (2005) and Koehn et al. (2003), showed that features focused in SMT are not helpful to track the long-distance dependency of a sentence like in Recurrent Neural Network. Collobert and Weston (2008) explained, how semantic, synthetic and morphological similarities are captured better when there is continuous representation of words that carry task-dependent knowledge.

Kalchbrenner and Blunsom (2013) introduced the Recurrent Continues Translation model, which has two parts, Convolutional Sentence Model and Recurrent Language Model. Convolutional Sentence Model, uses a convolutional n-gram approach on source sentences in the encoder. The sentences are mapped into semantic vectors. The recurrent language model is applied in the decoder. A similar approach is proposed by Cho et al. (2014), where the Recurrent Neural Network (RNN) is used for translation. Sentences needing translation are encoded into a sequence with a fixed length vector and decoded with another sequence of symbols. In this approach, the encoder and the decoder are jointly trained to increase the conditional probability of phrase pairs in the sentence using RNN.

Some research studies on translation are based on a monolingual dataset. A semi-supervised approach is proposed by Cheng and Duan (2020) with labelled and unlabeled corpus. Labelled corpus is a parallel corpus with source and target sentences of the Chinese-English dataset. Unlabeled corpus contains the monolingual dataset. In the semi-supervised setting, parallel and monolingual corpus are joined to learn Bidirectional NMT (source to target and target to source models). Sennrich et al. (2015) proposed two approaches to translate monolingual datasets. In the first approach the monolingual corpus is matched with dummy inputs to construct the parameters of the encoder with attention model (Choi et al., 2018) and the second approach utilizes a pre-trained NMT model.

Using NMT for translating one standard language to another standard language has been a success over the decade (Sreelekha et al., 2016). However, it is not experimented well with the code-mixed text due to the lack of resources. In the domain of translating code-mixed text, very few researches have been carried on. A combined approach of Statistical Modelling (Neale et al., 1999) and Knowledge Translation (Rijhwani et al., 2007) approach is introduced by Carrera et al. (2009) for cross-language social media texts. Rijhwani et al. (2016) introduce an approach for translating code-mixed text, where words in sentences are categorized as dominant and non-dominant languages. Words from dominant language are labelled as matrix language and non-dominant language are labelled as embedded language. The first task before the translation was word-level language identification. Next, the data is applied to a current translator to translate the words to another language. Dhar et al. (2018) used a code-mixed corpus from ICON 2017 tool contest for translation. Machine translation augmentation approach was used in their research study which achieved a BLEU score of 16.90. Masoud et al. (2019) used a combined corpus from OPUS3 and EnTan4, evaluated the corpus using several approaches and calculated the BLEU score. Word Hybrid Baseline approach achieved a BLEU score of 21.05, Byte Pair Encoding (BPE) Hybrid baseline approach achieved a BLEU score of 21.93, Word Hybrid Baseline Google approach received a BLEU score of 21.35. Finally, the Word Hybrid Baseline Google approach achieved the
highest BLEU score of 22.46.

3 SECM text and corpus creation

Sinhala is the native language of the majority of the people in Sri Lanka. But in Sri Lankan social media, Sinhala-English code-mixed text is used frequently because of the multi-lingual users. In 2001 International Organization for Standardization published the ISO15919 standard. It is an international standard for romanization which includes many languages including Sinhala language. Weerasinghe et al. (2005) used the IPA (International Phonetic Alphabet) format to represent Sinhala letters in their research study. ISO15919 and IPA both present Sinhala letters with English alphabets, which is called transliterated format or romanized text format (Hettige and Karunananda, 2007). Wasala et al. (2006) propose a conventional tag set, which uses the 26 alphabets of English to present the phonetic sound of Sinhala letters using the festival framework. Code mixing is described as a way of writing roman script (Davies and Bentahila, 2007). Even though the standard tag sets from Punchimudiyanse and Meegama (2015) is defined as roman representation, the romanization of actual code-mixing used by multilingual societies is different from the standard tag sets. Figure 3 and Figure 4 shows us how the standard romanization defined for Sinhala letters differs from the roman representation used in Singlish text.

According to Figure 3 and Figure 4, Phonetic Tagset (PT) and Romanized Tagset (RT) are almost similar. But there is a huge difference between these two tagset and code-mixed text representations of Sinhala letters. Due to no consistency in the pattern of Singlish text, it is not easy to translate the Sinhala code-mixed text without a parallel corpus. To achieve a good outcome most machine translation systems needs a sufficient amount of parallel sentences in the corpus.

We collected the SECM sentences from public Facebook posts, comments and reviews. For the translation of SECM to Sinhala, implementing the parallel corpus is an important part for our research study. Each SECM sentence was human translated by a linguistic expert of Sinhala language to create the parallel corpus. In the parallel corpus SECM is considered the source sentence and Sinhala is considered the target sentence. The human translator was advised to follow the Singlish to Sinhala mapping provided in the research study of Kugathasan and Sumathipala (2020) as the guide to maintain the consistency in the translation. The corpus contains around 1500 parallel sentences of SECM and it’s translated Sinhala sentences. After the human translation the dataset was checked to see whether the translated sentences are FC (Fully Correct) or CR (Correction Required). If a parallel sentence was annotated with a Correction Required tag by the annotator, the same annotator would provide the alternate translation as well. Each sentence in the corpus is annotated by two annotators. The annotators are people whose native language is Sinhala and we made sure that they are fluent in the Sinhala language. The annotators were provided with guidelines regarding the annotation process. The guidelines made sure that the annotators were checking whether there are any

![Figure 2: Example of SECM code-mixed text.](image-url)
spelling mistakes, grammatical issues in the Sinhala translation.

As shown in Figure 5, when there is a sentence tagged with two different tags, for example one annotator annotated with FC tag and the other annotated with CR tag with an alternate translation, we considered the alternate translation. Also when a sentence is annotated with CR tag from both the annotators, the alternate translations provided by both the reviewers are checked with a third annotator and the most suitable translation is selected. To evaluate the standard of the corpus, 100 randomly chosen sentences are provided to 3 experts of Sinhala for ranking. The translations are ranked as good or bad, according to the meaningful translation, grammatical pattern and spelling errors. Fleiss’ Kappa approach is used to calculate the reliability of the agreement between the raters (Randolph, 2008). The overall Fleiss’ Kappa score received for the translation is 0.88.

4 Methodology

After corpus creation, the dataset was applied with several pre-processing steps. Initially, all the sentences in the Singlish corpus are converted into lower case and all the Sinhala sentences in the target corpus are added with START and END tokens. The unique words from the corpus are extracted and unique numbers are allocated for each word according to the order of frequency of each word. These word-number arrays are called WordToIndex dictionaries.

Encoder-Decoder framework is used as the base to initiate our model. Encoder and decoder can be considered as two separate Recurrent Neural Networks. In the encoder the SECM sentence is fed as input. It produces the sentence with fixed-sized representation by encoding. Each word from the input sentence provided into the encoder would be mapped into an integer using the WordToIndex dictionary and converted into one-hot encoding. The embedding layer maps the one-hot encoded representation into a smaller dimension. The word embedding would be the input to the next layer with Long Short Term Memory (LSTM) as the basic unit. In LSTM we have a cell state that is passed with each timestep. The LSTM unit (Sundermeyer et al., 2012) determines to neglect some unnecessary information and add some new information from the input fed to the current timestep.

The significant information collected from the encoder would be passed into a context vector with the output and hidden states. Only hidden states are passed as input to the decoder.

Decoder produces the target sentence using the significant information passed through the hidden state from the encoder and the input target word. Each word from the target sentence is mapped into an integer using Word-
ToIndex dictionary and converted to one-hot encoding. The word embedding layer maps the embedding into a continuous representation which has a lot smaller dimension. Figure 6 shows the architecture inside timesteps in encoder and decoder.

Teacher Forcing algorithm (Goodfellow et al., 2017) is added in the decoder. Teacher Forcing algorithm inputs the expected output of previous timestep $t-1$ to the current timestep $t$. The advantage of using Teacher Forcing mechanism is the hidden state of the model would be updated with the correct expected outputs rather than the wrongly predicted output from the previous timestep. If the predicted output from previous timestep $t-1$ is fed to the next timestep $t$, the number of errors would be increased and the model would face difficulty in learning. Combination of encoder and decoder is called as Sequence to Sequence model (Seq2Seq) as shown in Figure 7.

Final phase of the proposed architecture of the system is prediction. Singlish sentence from the corpus is given as input, and output would be the predicted Sinhala sentence. Prediction phase is built with the sequence to sequence architecture. Each timestep in decoder passes predicted output for the next timestep unlike the decoder in the training phase of the model.

5 Experimental setting and Result

From the corpus 70% of the data is allocated for training, and 30% of the data is allocated for testing. Inputs for the encoder and the decoder are in the shape of 2D arrays. The shape of the encoder array is (10,27), where the batch size is ten and maximum length of source sentence is twenty seven. Shape of the decoder array is (10,26), where the batch size is ten and maximum length of source sentence is twenty six. Rmsprop is used as the optimizer and Categorical Cross Entropy is used to calculate the loss. The RMSprop optimizer is used because it balances the step size and, decreases the no of steps for massive gradients to neglect the exploding and increases the number of steps for small gradient to avoid vanishing gradients issue. Weights calculated after the training phase of the model are saved for the prediction phase. The model reached the training accuracy of 71.42% and testing accuracy of 37.17%.

After the model’s training, randomly se-

---

**Figure 5:** Sample sentences from annotated corpus, A1 - Annotator1, A2 - Annotator2, A3 - Annotator3, FC - Fully Correct, CR - Correction Required, N/A - Not Applicable

**Figure 6:** Architecture of each timestep in Encoder and Decoder

**Figure 7:** Encoder - Decoder framework
lected hundred SECM sentences from the corpus are inputted to predict the translated Sinhala sentence. The predicted Sinhala sentences are saved to calculate the BLEU score. Figure 8 shows examples of some predicted Sinhala translations. BLEU score is the evaluation metrics \citep{Papineni:2002:ACL:1073083.1073089} used to evaluate the translated sentences. BLEU metric provides a score for the translation based on the predicted sentence and relevant reference sentence. For each sentence in the corpus, modified precision of unigram, bigram, trigram and four-gram are calculated. The weight of 0.25 has been given to each modified precision.

\[
BLEU = BP.\exp\left(\sum_{n=1}^{N} W_n \log p_n\right)
\]

Equation 1, is used to calculate the BLEU score. \(N\) is the number of n-grams and \(W_n\) is the weight for each modified precision, \(p_n\) is modified precision. BP is the brevity penalty to penalize short machine translations\citep{Papineni:2002:ACL:1073083.1073089}.

\[
BP = \begin{cases} 
1 & \text{if } c > r \\
\exp(1 - \frac{r}{c}) & \text{if } c \leq r
\end{cases}
\]

The value of \(BP\) is decided according to the values of \(c\) and \(r\). \(c\) is the number of unigrams in all the predicted sentences, and \(r\) is the best match length for each predicted sentence in the corpus. Our model received a cumulative BLEU4 score of 31.54. Comparing to previous models proposed for code-mixed text translation \citep{Dhar:2018:1809.00742,Masoud:2019:1905.03364} our proposed approach with Teacher Forcing mechanism gives a remarkable BLEU score for the translation.

\[
\begin{array}{|l|l|l|l|}
\hline
\text{Sinhala Sentence} & \text{Reference Sentence} & \text{Predicted Sentence} & \text{BLEU} \\
\hline
dawalta crowd eka wadi & ප්‍රියඩු කොම්ඩු කවුඹ දසිලක් & සම්බන්ධ කවුඹ කවුඹ දසිලක් & 8.64E-78 \\
gali gandak enawa restaurant & මැසි දැන් මැසි දැන් මැසිය විසිදුම් දසිලක් & දසිලක් දසිලක් දසිලක් දසිලක් & 9.85E-232 \\
godaak expensive food, & මැසිබුදු කොම්ඩු කවුඹ කවුඹ කවුඹ & කවුඹ කවුඹ කවුඹ & 1 \\
 godaak senaga wadi & කවුඹ කවුඹ කවුඹ කවුඹ & කවුඹ කවුඹ කවුඹ කවුඹ & 1 \\
ehitharam not good & මැසිබුදු කවුඹ කවුඳු කවුඳු & මැසිබුදු කවුඳු & 1.38E-231 \\
\hline
\end{array}
\]

Figure 8: Example of predicted sentences and relevant BLEU score. Words highlighted in red are the words that are different from the reference sentence.

6 Conclusion & Future work

This paper presents a deep analysis of Sinhala-English code mixed texts. The difference between standard tagsets available for the romanization of Sinhala letters and the romanization used in SECM text are compared. The differences are discussed in this study. Challenges in the pattern of SECM sentences such as code-switching, spelling errors, improper usage of discourse marker etc., are also discussed in this paper. A parallel corpus is created containing SECM sentences and the relevant Sinhala sentences translated by a human translator who is a linguistic expert. The corpus is validated using annotators who are native Sinhala language speakers. The parallel corpus introduced in this paper can be considered a useful resource for researches based on SECM text. We combined Teacher Forcing Algorithm with the Sequence to Sequence approach with LSTM units to translate SECM sentences to Sinhala sentences. Teacher Forcing Algorithm updates the hidden state of each timestep in the decoder with the expected output from the previous time step, which leads on providing more accurate results for the translation. The BLEU score received for our model revealed that comparing the state of the art of other translation models for code-mixed texts, our model achieved significantly higher BLEU score. The future work we would like extend this research to focus on sentiment analysis and entity extraction using the parallel corpus created in this research study.

References

Jaime Arguello, Jonathan L Elsas, Jamie Callan, and Jaime G Carbonell. 2008. Document repre-
sentation and query expansion models for blog recommendation. ICWSM, 2008(0):1.

Jordi Carrera, Olga Beregovaya, and Alex Yani- shevsky. 2009. Machine translation for cross-
language social media. PROMT Americas Inc.

Yong Cheng and Mofan Duan. 2020. Chinese gram-
matical error detection based on BERT model. In Proceedings of the 6th Workshop on Natural 
Language Processing Techniques for Educational 
Association for Computational Linguistics.

model for statistical machine translation. In 
Proceedings of the 43rd annual meeting of the 
association for computational linguistics (acl’05), pages 263–270.

Kyunghyun Cho, Bart Van Merriënboer, Caglar 
Gulcehre, Dzmitry Bahdanau, Fethi Bougares, 
Holger Schwenk, and Yoshua Bengio. 2014. 
Learning phrase representations using rnn 
encoder-decoder for statistical machine transla-

Heeyoul Choi, Kyunghyun Cho, and Yoshua Bengi-
io. 2018. Fine-grained attention mechanism 
for neural machine translation. Neurocompu-
ing, 284:171–176.

Ronan Collobert and Jason Weston. 2008. A uni-
ified architecture for natural language processing: 

Eirlys E Davies and Abdelali Bentahila. 2007. Con-
tact linguistics: Bilingual encounters and gram-
matical outcomes.

Mrinal Dhar, Vaibhav Kumar, and Manish Shri-
vastava. 2018. Enabling code-mixed translation: 
Parallel corpus creation and MT augmentation 
approach. In Proceedings of the First Workshop 
on Linguistic Resources for Natural Language 
Processing, pages 131–140, Santa Fe, New Mex-
ico, USA. Association for Computational Lin-
guistics.

Ian Goodfellow, Yoshua Bengio, and Aaron 

Budditha Hettige and Asoka S Karunananda. 2007. 
Transliteration system for english to sinhala ma-
chine translation. In 2007 International Con-
ference on Industrial and Information Systems, 
pages 209–214. IEEE.

Nal Kalchbrenner and Phil Blunsom. 2013. Recur-
rent continuous translation models. In Proceed-
ings of the 2013 conference on empirical methods 
in natural language processing, pages 1700–1709.

Philipp Koehn, Franz J Och, and Daniel Marcu. 
2003. Statistical phrase-based translation. Tech-
nical report, UNIVERSITY OF SOUTHERN 
CALIFORNIA MARINA DEL REY INFOR-
mation SCIENCES INST.

Arachchana Kugathasan and Sagara Sumathipala. 
2020. Standardizing sinhala code-mixed text us-
ing dictionary based approach. In 2020 Intern-
national Conference on Image Processing and 
Robotics (ICIP), pages 1–6. IEEE.

Maraim Masoud, Daniel Torregrosa, Paul Buite-
approach for code-switching machine transla-
tion: A case study. In Proceedings of the 27th 
AAAI Irish Conference on Artificial Intelligence 
and Cognitive Science, Galway, Ireland.

Michael C Neale, Steven M Boker, Gary Xie, and 
H Mx Maes. 1999. Statistical modeling. Richmond, 
VA: Department of Psychiatry, Virginia 
Commonwealth University.

Kishore Papineni, Salim Roukos, Todd Ward, and 
Wei-Jing Zhu. 2002. Bleu: a method for auto-
matic evaluation of machine translation. In Pro-
ceedings of the 40th annual meeting of the As-
sociation for Computational Linguistics, pages 
311–318.

M Punchimudiyunase and RGN Meegama. 2015. 
Unicode sinhala and phonetic english bi-
directional conversion for sinhala speech recog-
nizer. IEEE International Conference on Indus-
trial and Information Systems 2015.


Shruti Rijhwani, Royal Sequiera, Monojit Choud-
hury Choudhury, and Kalika Bali. 2016. Trans-
lating codemixed tweets: A language detection 
based system. In 3rd Workshop on Indian Lan-
guage Data Resource and Evaluation-WILDRE-
3, pages 81–82.

Rico Sennrich, Barry Haddow, and Alexandra 
Birch. 2015. Improving neural machine trans-
lation models with monolingual data. arXiv 

Bryan Sippel and Susan E Brodt. 2008. The psy-
chology of blogging communities: Social identi-
ties and knowledge transfer across work-groups. In ICWSM.

S Sreelekha, Pushpak Bhattacharyya, Shishir K 
Jha, and D Malathi. 2016. A survey report on 
evolution of machine translation. Int. J. Control 

Pinjai Sudsawad. 2007. Knowledge translation: 
introduction to models, strategies and mea-
sures. Southwest Educational Development Lab-
oratory, National Center for the ....


Multilingual Multi-Domain NMT for Indian Languages

Sourav Kumar
IIIT Hyderabad
sourav.kumar@research.iiit.ac.in

Salil Aggarwal
IIIT Hyderabad
salil.aggarwal@research.iiit.ac.in

Dipt Misra
IIIT Hyderabad
dipti@iiit.ac.in

Abstract

India is known as the land of many tongues and dialects. Neural machine translation (NMT) is the current state-of-the-art approach for machine translation (MT) but performs better only with large datasets which Indian languages usually lack, making this approach infeasible. So, in this paper, we address the problem of data scarcity by efficiently training multilingual and multilingual multi domain NMT systems involving languages of the Indian subcontinent. We are proposing the technique for using the joint domain and language tags in a multilingual setup. We draw three major conclusions from our experiments: (i) Training a multilingual system via exploiting lexical similarity based on language family helps in achieving an overall average improvement of 3.25 BLEU points over bilingual baselines, (ii) Technique of incorporating domain information into the language tokens helps multilingual multi-domain system in getting a significant average improvement of 6 BLEU points over the baselines, (iii) Multistage fine-tuning further helps in getting an improvement of 1-1.5 BLEU points for the language pair of interest.

1 Introduction

Good translation systems are an important requirement due to substantial government, business and social communication among people speaking different languages. Neural machine translation (Vaswani et al., 2017) is the current state-of-the-art approach for machine translation in both academia and industry (Bahdanau et al., 2014). The success of NMT heavily relies on a huge amount of parallel sentences as training data. But using the traditional approaches (Vaswani et al., 2017), one would still need to train a separate model for each translation direction and also a lot of parallel human translated corpora which is again expensive to generate. But on the other hand, multilingual neural machine translation (Johnson et al., 2017) enables training a single model that supports translation from multiple source languages to a single target language or from a single source language to multiple target languages. In addition to this, another benefit of a training a single model for multiple translation directions is the ability to learn not just from the training data of the language pair of interest, but also from other language pairs. But this learning is hindered in case of language pairs that do not show any kind of relatedness among themselves. But on the other hand, Indian languages exhibit a lot of lexical and structural similarities on account of sharing a common ancestry. It is therefore important to utilize the lexical similarity (Kunchukuttan and Bhattacharyya, 2020) of these languages to build efficient systems by combining all the related languages.

Also, in a typical in-domain MT scenario, the amount of parallel texts from a single domain is not enough to train a good translation system, even for multilingual systems. Apart from this, one has to train an individual MNMT system for each domain. So we propose a technique for creating efficient multilingual multi-domain NMT systems which help in overcoming the above mentioned limitations. In this work, we treat different domains as distinct languages: for example, instead of Hindi-English translation we see it as translating Hindi-health to English-health. We utilized our multilingual NMT approach in a multi-domain setting and our results confirm that our multilingual multi-domain system significantly outperforms in-domain baselines as well as it also give improvement for out-of-domain translations. The paper is organized as follows: Section 2 talks about the related work. Methodology for our
experiments is explained in Section 3, followed by experimental details and results in Section 4 and Section 5 respectively. All the conclusions and the future work have been briefly discussed in Section 6.

2 Related Work

Due to simplicity, generality and effectiveness of Neural Machine Translation (NMT), it has become the most prominent approach to machine translation (Luong et al., 2015; Bahdanau et al., 2014; Johnson et al., 2017; Wu et al., 2018; Vaswani et al., 2017). Basic training procedure of NMT does not work well with only a handful of bilingual data (Koehn and Knowles, 2017), while collecting bilingual resources is arduous for Indian languages. A lot of experiments have been made to improve the quality of translation mainly including exploiting monolingual data range from back translation (Sennrich et al., 2015), dual learning (Xia et al., 2016) to Unsupervised MT (Artetxe et al., 2017; Lample et al., 2017). On the other hand, many tried to exploit parallel data of other high resource languages (Zoph et al., 2016; Firat et al., 2017; Johnson et al., 2017; Kocmi and Bojar, 2018) to either pre-train the network or jointly learn the representation.

Recently multilingual NMT has drawn more attention by several research groups. For instance, Firat et al. (2016) modify the current state-of-the-art attention NMT approach by introducing a many-to-many system, which still relied upon separate encoders and decoders for each language along with a shared attention mechanism. In contrast, Johnson et al. (2017) and Ha et al. (2016) both introduce a simple method for training a single-model multilingual NMT system, which does not require any modifications to the NMT encoder-decoder architecture of the system. The main difference is that Johnson et al. (2017) added target language identifying token in the beginning of each source sentence of the training data and Ha et al. (2016) added a language identifying token to each subword unit and apply this pre-processing to both source and target sentences of the training data. Both aims at exploiting many different languages rather than focusing on language relatedness and observes that only the many-to-one paradigm can achieve better translation results than the individually trained models. For the other two paradigms, there are various degrees of quality degradation. Also Vertan and von Hahn (2013) has put some efforts in tackling efficient NMT system in low-resource settings by considering language relatedness.

Apart from this, researchers have also explored the area of domain adaptation for NMT (Chu and Wang, 2018) and reported significant improvements. (Tars and Fishel, 2018; Kobus et al., 2016) explored multi-domain Neural machine translation for single language by adding the domain token to the input sentence. So, we put our efforts in exploring the system performance when multilingualism is combined with multi-domain systems for major Indian languages.

3 Methodology

India is a land of diverse languages. It has many languages on the basis of regional diversities, and mainly divided into Indo-Aryan and Dravidian families. A universal characteristic of Indian languages is their complex morphology. Indian languages depict unique characteristics following default sentence structure as subject object verb (SOV) and relatively free word order. These languages also share many common words which have the same root and meaning. However they use different scripts derived from the ancient Brahmi script (Kunchukuttan and Bhattacharyya, 2020), but correspondences can be established between equivalent characters across scripts. So, we exploited lexical similarity for efficient MNMT. Also, in order to incorporate multiple domains into our multilingual system, we also introduced the technique of representing the domain as a new language.

3.1 Exploiting Lexical Similarity

Unlike the original multilingual NMT (Johnson et al., 2017) which aims at exploiting many different languages rather than focusing on language similarity. Thus, to exploit the language relatedness we efficiently combined the two different approaches namely Unified Transliteration and Subword Segmentation to ensure that there is a sufficient overlap between the vocabularies of the related languages.

3.1.1 Unified Transliteration

Since the languages involved in the models have different orthographies and relatedness among each
other, also Indo-Aryan and Dravidian family languages don’t share many common characteristics, thus in the data processing one can map these two different language families into two different common orthographies. In order to achieve this, we transliterated all the Indian language into a common script based upon their family groups using the Indic NLP library (Kunchukuttan, 2020). Indo-Aryan languages were transliterated to Hindi (Devnagri script) while Dravidian languages were transliterated to Tamil (Abugida script) to share the same surface form within their family. This unified transliteration is a string homomorphism, replacing characters in all the languages to the desired script.

3.1.2 Subword Segmentation
Most of the Indian languages are derived from the common ancient Brahmi script and share many common words at the root level. To do so, we used Byte Pair Encoding (BPE) (Sennrich et al., 2015) to break words into subwords. Also, BPE merge rules not only find the common subwords between two related languages but it also ensures consistency of segmentation among each considered language pair. We are learning the BPE rules by combining all the languages on the source and the target side respectively, thus further applying these rules for segmenting the corpora. This finally results in increasing the vocabularies overlap among the languages that we made share the same surface form by transliterating into a common desired script.

3.2 Multilingual and Multi-Domain Systems
Multilingual model enables us to translate to and from multiple languages using a shared word piece vocabulary, which is significantly simpler than training a different model for each language pair. Johnson et al. (2017) introduced a “language flag” based approach that shares the attention mechanism and a single encoder-decoder network to enable multilingual models. A language flag or token is part of the input sequence to indicate which direction to translate to. The decoder learns to generate the target given this input. This approach has been shown to be simple, effective and forces the model to generalize across language boundaries during training. It is also observed that when language pairs with little available data and language pairs with abundant data are mixed into a single model, translation quality on the low resource language pair is significantly improved.

Multi-domain model is a single model that supports multiple domains in one model and also allows switching between the domains when translating. Similar to Johnson et al. (2017), Tars and Fishel (2018) explored Multi-Domain Neural Machine Translation for single language by adding the domain token to the input sentence instead of the language token.

3.3 Multilingual Multi-Domain Systems
A lot of research areas have been explored separately for multilingual as well as Multi domain systems. But no work has been done on combining both these approaches for the Indian languages. To the best of our knowledge, this is the first time efforts have been made in combining these two techniques for the NMT systems trained for Indian languages. So in this paper, we present our
technique of training a multilingual multi-domain system using the same traditional encoder-decoder architecture with shared attention mechanism. To do so, we introduced a **special token** technique which incorporates the knowledge of the language as well as the domain. The core idea is to treat ‘domains as distinct languages’ while training multilingual multi-domain systems. The pipeline for our technique can be seen in Figure 1.

For efficiently exploiting the language relatedness, we are first identifying and grouping the languages based on their lexical similarities with each other. In our case of Indian languages, all of the languages were first divided into 2 groups namely Indo Aryan and Dravidian. For each group, we appended our special token \((\text{2Lan-Domain})\) for one to many systems and \((\text{Domain})\) token for many to one systems. Then each group is being transliterated to a common script. For our purpose, Indo aryan languages were transliterated into Hindi while Dravidian languages were transliterated into Tamil. Then to increase vocabulary overlap amongst related languages at root word level, we are using BPE as discussed in Section 3.2. Further, we used this model learning for multistage fine tuning.

### 3.3.1 Multistage Fine Tuning

In the normal transfer learning Zoph et al. (2016) approach for NMT, the parent model is trained on a single high-resource language pair which may or may not be related to the child language pair of interest. To the best of our knowledge, previous transfer learning approaches for Indian languages do not exploit parallel data from multiple languages and domains. However, learning from multiple languages and domains can result in better knowledge transfer. Therefore, in this work, we propose a new transfer learning approach called as ‘**Multistage Transfer Learning**’ to enable the low-resource languages to efficiently learn from multiple related languages as well as domains which may or may not be high-resourced. In this approach, the parent model is our multilingual multi-domain NMT system and after pre training the parent model, the child model is initialized with parent model parameters and is then fine-tuned multiple times.

The proposed approach delivers better results than multilingual multi-domain NMT because adding more languages and domains into one model may result in better knowledge transfer but it can also result in ambiguities between different languages and domains at the inference time. Accordingly, a multilingual multi-domain NMT system fine-tuned can potentially remove all the inconsistencies at the inference time. For the scope of this paper, we have performed multistage fine tuning in three different scenarios: (i) single domain multiple language, (ii) multiple domain single language and (iii) single domain single language.

### 4 Experimental Settings

#### 4.1 Dataset

In our experiments, we are using the multi parallel corpus of two completely different domains namely PMI (Prime Minister of India) (Haddow and Kirefu, 2020) which contains the news domain aligned sentences and ILCI (Indian Language Corpora Initiative) (Jha, 2010) which is a combination of health as well as tourism domain sentences. ILCI corpus contains translation of the same sentence in every language pair but PMI contains roughly 60% of common sentences being translated to all language pairs, where Hindi being the high resource language.

#### 4.2 Data Preprocessing

We also noticed the ILCI corpus contains a lot of disalignments and empty translations, so we put our efforts in cleaning the entire corpus maintaining multi parallelism. We ended up in removing 3099 sentences from the corpus. Training data set statistics of both the data sets are mentioned in Table 1. All of our experiments were tested on 1870 sentences from ILCI and 2390 from the PMI corpus with validation data of 500 from both. We also made sure that there is no overlap between the

<table>
<thead>
<tr>
<th>Data</th>
<th>En-hi</th>
<th>En-pa</th>
<th>En-gu</th>
<th>En-mr</th>
<th>En-bn</th>
<th>En-or</th>
<th>En-kn</th>
<th>En-ml</th>
<th>En-ta</th>
<th>En-te</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMI</td>
<td>50349</td>
<td>28294</td>
<td>41578</td>
<td>28794</td>
<td>23306</td>
<td>31966</td>
<td>28901</td>
<td>26916</td>
<td>32638</td>
<td>33380</td>
</tr>
<tr>
<td>ILCI</td>
<td>44601</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Except or and kn; Include ko)

Table 1: Training Dataset Statistics
Table 2: PMI results (En-XX)

<table>
<thead>
<tr>
<th>Type</th>
<th>En-hi</th>
<th>En-pa</th>
<th>En-gu</th>
<th>En-mr</th>
<th>En-bn</th>
<th>En-or</th>
<th>En-kn</th>
<th>En-ml</th>
<th>En-ta</th>
<th>En-te</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bilingual Baseline</td>
<td>23.21</td>
<td>18.26</td>
<td>15.46</td>
<td>7.07</td>
<td>5.25</td>
<td>8.32</td>
<td>8.67</td>
<td>4.83</td>
<td>5.32</td>
<td>6.12</td>
</tr>
<tr>
<td>Multi Domain Bilingual Baseline</td>
<td>24.48</td>
<td>19.41</td>
<td>16.23</td>
<td>8.37</td>
<td>6.36</td>
<td>-</td>
<td>-</td>
<td>4.86</td>
<td>6.09</td>
<td>6.75</td>
</tr>
<tr>
<td>Multilingual Multi Domain with Token</td>
<td>29.29</td>
<td>27.29</td>
<td>21.86</td>
<td>14.57</td>
<td>11.04</td>
<td>15.53</td>
<td>15.79</td>
<td>10.16</td>
<td>9.88</td>
<td>8.67</td>
</tr>
</tbody>
</table>

Table 3: ILCI results (En-XX)

<table>
<thead>
<tr>
<th>Type</th>
<th>En-hi</th>
<th>En-ur</th>
<th>En-pa</th>
<th>En-gu</th>
<th>En-mr</th>
<th>En-bn</th>
<th>En-kn</th>
<th>En-ml</th>
<th>En-ta</th>
<th>En-te</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bilingual Baseline</td>
<td>22.65</td>
<td>21.01</td>
<td>18.60</td>
<td>14.45</td>
<td>10.08</td>
<td>12.15</td>
<td>8.28</td>
<td>3.29</td>
<td>1.97</td>
<td>5.06</td>
</tr>
<tr>
<td>Multi Domain Bilingual Baseline</td>
<td>23.64</td>
<td>-</td>
<td>19.52</td>
<td>16.48</td>
<td>10.95</td>
<td>13.79</td>
<td>-</td>
<td>4.42</td>
<td>3.04</td>
<td>6.23</td>
</tr>
<tr>
<td>Multilingual</td>
<td>27.02</td>
<td>22.32</td>
<td>23.08</td>
<td>21.5</td>
<td>14.08</td>
<td>16.79</td>
<td>14.41</td>
<td>6.19</td>
<td>5.18</td>
<td>9.36</td>
</tr>
<tr>
<td>Multilingual Multi Domain with Token</td>
<td>29.26</td>
<td>26.47</td>
<td>24.66</td>
<td>23.23</td>
<td>15.34</td>
<td>18.12</td>
<td>15.46</td>
<td>6.72</td>
<td>5.87</td>
<td>10.58</td>
</tr>
</tbody>
</table>

designed test and training set of PMI corpus. We used the Moses (Koehn et al., 2007) toolkit for tokenization and cleaning the English side of the data and we used the Indic NLP library (Kunchukuttan, 2020) for the normalization, tokenization and transliteration for the Indian languages. In all cases, we used BPE segmentation with 12k merge operations as described in Section 3.1.2

4.3 Training and Evaluation Details

For all of our experiments, we use the OpenNMT-py (Klein et al., 2017) toolkit. We used the Transformer model with 6 layers in both the encoder and decoder, each with 512 hidden units. The word embedding size is set to 512 with 8 heads. The training is done in batches of maximum 4096 tokens at a time with dropout set to 0.3. We use the Adam (Kingma and Ba, 2014) optimizer to optimize model parameters. We validate the model every 5,000 steps via BLEU (Papineni et al., 2002) and perplexity on the development set. We are training all of our NMT models with early stopping criteria based on validation set accuracy. During testing, we rejoin the translated BPE segments and convert the translated sentences back to their original language scripts. Finally, we evaluate the accuracy of our translation models using BLEU.

5 Results and Analysis

We report the results of bilingual baseline, multi-domain bilingual baseline, multilingual, multilingual multi-domain and multilingual multi-domain with special tokens for both the translation directions, XX-En and En-XX (where XX denotes Indian Languages). Later, we also compared the results of multistage fine tuning with the above experiments for one language from each family.

Table 2 and 3 shows our main results for English to Indian languages (En-XX) translation direction for PMI and ILCI corpus respectively. In both the cases, we observed that our multilingual model shows significant improvements over the baseline, increasing average BLEU score of 5 and 4 respectively. The reason behind this is that in the En-XX direction, language flags are used on the source side which then helps the decoder to identify the direction it translates to.

Table 4 and 5 shows our main results for Indian languages to English (XX-En) translation direction for PMI and ILCI corpus respectively. In the case of the ILCI dataset, we do not observe any significant improvements. The reason for this might be the multi parallel nature of the ILCI dataset where each English sentence on the target side appears multiple times in the model, thereby creating ambiguities in the model. But for the case of the PMI dataset, we observed an average improvement of 6 BLEU points, mainly due to large improvements in low resource languages. The reason for the increase in BLEU score for PMI is that the distribution of data is not uniform. Some of the languages in PMI corpus are low-resource as compared to others thus allowing other high resource languages to assist in the learning process for low resource languages thus removing the ambiguity. We also showed that the results of the multilingual multi-domain system with our special language domain outperforms the without token case for both the domains giving
an average improvement of \textbf{1.5 Bleu points} in the En-XX direction and \textbf{4 BLEU points} in XX-En direction over a normal multilingual system. We also experimented the Multistage Fine tuning for Punjabi and Tamil following the above three different scenarios mentioned above in Section 3.3.1 and observed an improvement within \textbf{1 - 1.5 BLEU points} over the multilingual multi-domain system.

6 Conclusions and Future Work

In this paper, we explored different effective methods to exploit parallel data from multiple related languages and domains to improve the translation between Indian languages and English. Our results show that the multilingual models accuracy depends upon the type of dataset in hand. As we observed in the case of PMI and ILCI, multilingual models trained on the PMI dataset increased our average BLEU score while the model trained on the ILCI dataset decreased the BLEU score due to increase in ambiguity. We then introduced multilingual multi-domain models and observed that this idea helps in removing the ambiguity we faced in the multilingual system using multi parallel data for training thus improving the translation quality by showing improvements in BLEU scores. In this work, we also introduced a new technique of adding a domain as a separate language by modifying the language token to language domain token. Our experiments also confirm that this new technique always outperforms all the models we discussed above. At last, we also explored the concept of Multistage Fine Tuning in which we transfer the learning of the parent model to the child in multiple stages. In future, we would like to work on effective techniques to exploit monolingual data and parallel data from other languages together to improve the translation quality. Also, we will try to generalize this idea of exploiting the related languages to other NLP related applications like sentiment analysis.

References


Girish Nath Jha. 2010. The tdil program and the indian language corpora intitiative (ilci). In \textit{LREC}.


Fiction in Russian Translation: A Translationese Study

Maria Kunilovskaya¹, Ekaterina Lapshinova-Koltunski², and Ruslan Mitkov³

¹³University of Wolverhampton
²Saarland University
¹³maria.kunilovskaya@wlv.ac.uk, r.mitkov@wlv.ac.uk
²e.lapshinova@mx.uni-saarland.de

Abstract

This paper presents a translationese study based on the parallel data from the Russian National Corpus (RNC). We explored differences between literary texts originally authored in Russian and fiction translated into Russian from 11 languages. The texts are represented with frequency-based features that capture structural and lexical properties of language. Binary classification results indicate that literary translations can be distinguished from non-translations with an accuracy ranging from 82 to 92% depending on the source language and feature set. Multiclass classification confirms that translations from distant languages are more distinct from non-translations than translations from languages that are typologically close to Russian. It also demonstrates that translations from same-family source languages share translationese properties. Structural features return more consistent results than features relying on external resources and capturing lexical properties of texts in both translationese detection and source language identification tasks.

1 Introduction

This paper reports results of a translationese study based on literary texts translated from 11 source languages into Russian and included into the Russian National Corpus¹. Translationese is understood as specific linguistic properties of translations distinguishing them from non-translated language. Most existing investigations into the properties of translations have focused text types other than translation and, to the best of our knowledge, there were no large-scale investigations into translational Russian, especially based on literary texts. The research applications of the parallel multilingual corpus of translations into Russian are limited to contrastive studies such as (Dobrovolskij and Pöppel, 2017) and translation studies such as (Krasnopeyeva, 2016), and deal with a few individual items or constructions. We explore linguistic differences between fiction translated into Russian and originally-authored literature in Russian with a focus on the impact of the source language (SL).

The comparison between translated and non-translated Russian is based on an extended feature set capturing structural and abstract lexical parameters of texts, including collocational properties. This investigation mostly relies on the text classification approach and univariate statistical analyses following a methodology established in computational studies of translationese (Baroni and Bernardini, 2006; Volansky et al., 2015; Evert and Neumann, 2017, amongst others). Translationese indicators that cut across all translated subcorpora (if any) or are specific to a particular language pair are identified through feature selection.

It does not come as a surprise that researchers are wary of using fiction for translationese studies. Fiction can be less homogeneous register because each literary work and author might display unique conceptual and linguistic properties reflecting a particular manner of artistic expression and idiolect. With literary translation established as a form of creative endeavour of its own standing and with the covert ‘domesticating’ translation paradigm accepted as a professional standard, little difference between translations and non-translations is to be expected in this register. This study is based on a carefully balanced and representative sample of literary texts from each language pair designed to reduce possible impact of author/translator idiolects on the results.

We aim to establish if and how Russian non-translated literary texts differ from literary translations from a variety of SLs (our first research question, RQ1).

Following a recent trend in translationese stud-
ies (Dutta Chowdhury et al., 2020; Bjerva et al., 2019; Rabinovich et al., 2017), we are interested in exploring the link between the amount of translationese and the SL involved. Translations from more distant languages were shown to return better classification results indicating greater differences from non-translations. Can we corroborate this finding on our data and features (our second research question, RQ2)?

Our results are relevant to typological and contrastive studies that use parallel data to draw conclusions about the properties of target language (TL) because we are highlighting that translations constitute a TL variety, significantly diverging from comparable non-translations. This study also contributes to a direction in translationese studies that seeks to establish links between SL-TL cross-linguistic distance and the amount of observed translationese as well as to the task of document-level SL detection.

2 Related Work

Translations vs non-translations Our work is related to studies showing that translated texts share linguistic features that make them distinct from non-translations (Gellerstam, 1986; Baker, 1993). These features have been useful in automatic classification of translated and non-translated texts (Baroni and Bernardini, 2006; Volantsky et al., 2015; Rubino et al., 2016; Kunilovskaya and Lapshinova-Koltunski, 2020). Some studies show that combinations of features perform better in machine learning settings for translationese-related tasks (Lynch and Vogel, 2012; Evert and Neumann, 2017; Sominsky and Wintner, 2019). This paper compares the effectiveness of two feature sets: morphosyntactic and lexical indicators. The former are count-based features extracted directly from automatically annotated research data. The latter are estimated using language models trained on a large, register-comparable TL corpus. They capture dissimilarity of translations with regard to the TL norm manifested in this additional TL resource.

Translationese in literary texts There are several corpus-based studies of literary translations. Most of them explore single features: passive constructions (Kolehmainen and Rionheimo, 2016) that-complementiser (Olohan, 2001), non-finite constructions, phrasal verbs, connectives (Kunilovskaya, 2017), keywords (Puurtinen, 2003), etc.

The only two works known to us that use machine learning to study translationese in literary texts include (Popescu, 2011) and (Lynch and Vogel, 2012). The first study explored the effectiveness of character 5-grams comparing originally-written English literature with translations from French and German. The second study is more similar to the design proposed in this study. It reports the results of SL detection based on English literary translations from Russian, German and French. A Support Vector Machine (SVM) 4-class classifier achieves the accuracy of 80% in the train-test scenario using a combined set of 50 features. The authors attempt the analysis of some of the top-ranking features, trying to link them to the SLs. In our work, we also employ multiclass classification to identify best translationese predictors. However, we rely on very different features and experiment on a twice bigger data from a wider range of SLs.

Tracing source languages Source language detection is based on the assumption that translations tend to retain features of the SLs (shining through effect, Teich, 2003). Rabinovich et al. (2017) showed that translations carried enough signal from SLs to restore the phylogenetic language tree. The authors experimented on English translations of the European parliamentary speeches in 17 SLs representing three language families. They noticed that misclassified instances were frequently assigned to genetically related languages. Translations from SLs with isomorphic structures displayed a tendency to share more translationese features. Another study by Bjerva et al. (2019) used three levels of syntactic abstraction to explore genetic, geographical and structural distances between SLs. In their results, structural similarities were a better predictor of similarities between languages than genetic ones. Dutta Chowdhury et al. (2020) used isomorphism between embedding spaces, hypothesising that the more isomorphism was detected between translations into English and non-translated English, the closer the source and target languages. They learned delexicalised multi-view representations – embeddings based on parts-of-speech (PoS) tags, lexical semantic tags, and conceptual-semantic tags from WordNet. However, these studies are based on EuroParl, a corpus of parliamentary speeches, which might be more homogeneous in style, more conventionalised as a text type and in terms of translation strategies than literary texts focused in this paper.
The relation between predictability of translated texts and the divergence between the TL and the SL in terms of morpho-syntax was analysed in a recent study by Nikolaev et al. (2020). The authors used Parallel Universal Dependencies to analyse 1000 sentences from the news domain and Wikipedia translated from English into eight languages. The results showed that translations from similar and distant languages were both predictable, but in different ways: structurally-similar SLs favoured the use of a narrower range of syntactic patterns limited to those shared by two languages, which constituted one type of translational specificity. In translations from highly-divergent languages, however, translators tended to produce non-idiomatic renditions, that were not recognised by models trained on the TL. Sominsky and Wintner (2019) also used the SL signal to detect translation direction. They found that the more distant were the source and the target languages, the higher the SL-detection results.

3 Methodology

3.1 Research Corpus

Our data comes from the parallel component of the RNC. It is a bidirectional corpus, which contains translations from and into Russian for 19 language pairs. Fiction is by far the most represented register, however Russian translations of newspaper texts from some SLs are also included. At the time of writing, parallel RNC includes Russian translations of fiction from 19 SLs, with the overall size of this translational material estimated at 36.6 million tokens. This corpus was sampled to include all language pairs with Russian as the TL that have at least five documents of lengths over 20k tokens produced by a unique combination of author and translator. The document size limit is intended to exclude short stories and retain only novellas and novels. In the absence of a genre annotation, this restriction maintains some comparability of our documents with regard to the type of literary work. The author/translator condition minimises the impact of individual writing style on our results. Another data selection constraint ensures that translations were produced within a time span of 100 years (1925–2020). We excluded document pairs where the author’s mother tongue was not the respective SL (e.g. Nabokov) and where the translations were done by the author (e.g. Vasil Bykov). This sampling frame leaves us with a corpus of 210 document pairs in 11 parallel subcorpora. At pre-processing stage, we discarded all sentence pairs with empty source or target as well as by-lines and headings.

To build a comparable collection of non-translated Russian fiction, we used the same sampling frame on the monolingual part of the RNC. We retained only the largest work by every author, deleted the works in Russian by bilingual authors (e.g. Nabokov) and novels explicitly marked as translations. These selection criteria yielded a corpus counting 439 documents (longer than 20K tokens), 33.8 M tokens (3.2M sentences) in total. Details on the distribution of document lengths in the samples for each SL and Russian reference sample are presented in Figure 1. This material was used for chunking (see below).

The SLs in the resulting collection are distributed among four language families in our collection: Romance (French, Spanish), Germanic (Swedish, English, German), Balto-Slavic (Baltic: Latvian, Slavic: Polish, Belarusian, Ukrainian, Bulgarian), Uralic (Finnish) based on linguistically motivated phylogenetic language tree (see Serva and Petroni, 2008). In total, we have 859 documents in our data, including parallel and monolingual components. To balance the data in terms of document size, we randomly selected 10 book with unique author-translator combinations. For Bulgarian, Spanish, French, Finnish and Latvian we had to do with fewer books to meet this restriction. After that, each literary work was chunked into portions of 100 consecutive sentences and 10 random chunks were extracted. This collection of text chunks was used in the experiments below.

3.2 Features for classification

The key component of this methodology is the features. We classify them into structural and abstract lexical features.

Structural features. The first subset includes 45 features extracted from Universal Dependencies (UD) annotations of the data (UD features). The values for UD features are normalised frequencies of various UD tags and their combinations, reflecting the morphological and syntactic structure of language. They are selected to capture the differences in the linguistic make-up of translations demonstrated in translationese studies for other language pairs and anticipated in out-of-English Russian translations in the practical translation text-
books based on the typological differences of English and Russian. It has been shown that translations usually have lower type-to-token ratio, higher sentence length and greater number of connectives in a number of language pairs. It is also expected that Russian translations from English would have higher frequencies of modal predicates, analytical passive forms, inflated frequencies of several types of pronouns due to typological differences. The values for most features are cumulative frequencies of all lemmas that belong to a word class or all forms that received a specific tag. We summarise the types of structural features in Table 2 in Appendix2.

The normalisation basis varies depending on the type of item, following the motivation in Evert and Neumann (2017): it is total text tokens for word classes; number of sentences for conjunctions, modal predicates; total verbs for verb forms; total number of dependencies for the select types of dependencies, etc. The values for discourse markers features are normalised cumulative frequencies of four semantic types of connectives and of epistemic markers (e.g. of course, probably, actually), extracted based on pre-defined lists of lemmas (183 items in total for connectives and 86 items for epistemic markers). The lists are informed by Russian grammars and special linguistic dictionaries. Though most items on the lists are set phrases, we allowed for possible lexical and structural variability during extraction. We also used positional heuristics and punctuation to disambiguate our items. The output of the extraction procedure was manually checked to exclude greedy matching.

Abstract lexical features. The second subset, counting 23 features, requires language models learnt from a separate (bigger) corpus resource of original Russian literature (LM features). We used the corpus described in Section 3.1, excluding the 10 random books used to get 100 chunks of non-translated Russian subcorpus. We used a 3-gram language model learnt on this corpus with KenLM library (Heafield, 2011) to generate average sentence perplexities and their standard deviation for each text chunk in our data. We hypothesise that this model will return higher perplexities depending on how unusual the sequences of lexical items are in the translated language. N-grams frequency lists (of orders 1, 2 and 3) from the same corpus were used to calculate the ratios of lemmas that belonged to the highest- and lowest-frequency quartiles of this list for each order of the n-grams cumulatively. Ratios of out-of-vocabulary (OOV) items was used as separate features for each n-gram order. These features were supposed to capture the overuse of the TL high-frequency items and, possibly, a higher ratio of OOV items. These features were inspired by feature-based quality estimation approaches used for machine translation (Specia et al., 2015). We

2The extraction code is available from https://github.com/kunilovskaya/translationese45

Figure 1: Distributions of document sizes, number of observations and unique translators by SL in the filtered research corpus.
also experimented with 10 collocational features that are assumed to capture various aspects of co-occurrence patterns in the data. To define these features we relied on the concept of coligrams, defined by Bestgen and Granger (2014) as n-grams with an association score above an arbitrary threshold. For association measures we used normalised pointwise mutual information (NPMI) and t-score to detect coligrams composed of less-frequent and more-frequent words respectively. For each association measure we trained a bigram model on the large reference corpus described above with the Phrases module from the Gensim library (Řehůřek and Sojka, 2010). Then, the model was applied to the chunks of text in the test corpus. In this approach the coligrams were detected and the average association scores across all coligrams in each chunk were produced based on the frequency statistics in the reference corpus. T-score measure was calculated using the formula from Gries (2010). For NPMI we relied on the Gensim inbuilt scorer (Bouma, 2009). In both measures 0 means independence of bigram components; NPMI lies in the [-1:: 1] range, while t-score bounds were experimentally established within [-11:: 9] scope. To train the models, we set the association score threshold to the lower bound of each metric. The bigram frequency threshold was set to 1 to score all bigrams in the 33-million reference corpus. While learning the phraser model, we allowed for intervening words from the functional word classes to access items like альфаА:анга (alpha::and::omega), пакт::о::ненападение (non::aggression::pact). In calculating feature values we relied on bigram/collgram types, not tokens. To sum up, the collocational features for each association measure extracted from each chunk in the research corpus include: (1) ratio of highly-associated coligrams to all bigrams (the cut-off for high association was set to recommended NPMI > 0.5 and t-score > 6); (2) ratio of negatively-associated coligrams to all bigrams; (3) ratio of all detected coligrams with the score > 0 to the total word count; (4) ratio of bigrams absent in the model to all bigrams in the test corpora; (5) average association score for all detected coligrams with the association score > 0; (6) standard deviation for the association scores in each text chunk.

We expect that translations would have a higher ratio of t-score-based highly-associated bigrams, and lower ratio of NPMI-based highly-associated bigrams than in the comparable subcorpus of non-translations. This hypothesis is based on the known properties of translation to prefer high-frequency items and on the known properties of the association measures: MI is known to favour sequences of low-frequency items, while t-score assigns higher scores to high-frequency items (Gries, 2010). The ratio of negatively associated bigrams and the ratio of bigrams not seen in the reference is aimed to capture less usual sequences which can be a sign of shining though or errors, including acceptability in the register (e.g. тяжела критика, крепкая основа).

Finally, we calculate the average association score for each text chunk to reflect general ‘collocatedness’ of translations and non-translations and standard deviation across all chunks in each source-language subcorpus. All lexical features were produced on lemmatised corpora, where proper names and their sequences were replaced with PROPN and all numbers were represented as XXX (e.g. Борис Николаевич Юрьев -> PROP, 1984 -> XXXXX). We also deleted all punctuation, except end-of-sentence marks.

The features were extracted into a table containing 1060 rows representing all text chunks in our experiments, labelled with SL, including ‘ru’ for chunks selected from the reference corpus and 68 features, including 45 UD-based (inc. list-based features), 11 ngram-based and 12 collgram-based ones. The features in the last two groups were extracted with reference to several LMs learnt from a reference corpus of non-translated fiction. The input feature values were z-transformed with the scikit-learn Standard Scaler.

### 4 Results and Discussion

To find out whether our features capture any differences between translations and non-translations in SL subcorpora and whether the scale of these differences is traceable to the typological group of the SL, we ran 11 binary classifications and a multiclass classification. In all experiments we used Support Vector Machine (SVM) algorithm with the default scikit-learn settings (C=1.0, kernel=‘rbf’, degree=3, gamma=‘scale’). For the five subcorpora where we have fewer observations than in the reference sample, we used class_weight=‘balanced’ option.

Table 1 presents the accuracy (acc) and macro F1-score (F1) results of the binary classifications.
for each translational subcorpus (represented by SL) against non-translations. The SLs are presented in two groups: Typologically distant and close languages (in relation to Russian). We also group the scores according to the feature sets and combinations: all 68 features, 45 UD features (structural features), 11 n-gram and 12 collogram (both abstract lexical features). The last two columns show the results expected by chance. This simple baseline was implemented as a random classifier which predicts classes with respect to the distribution of instances across classes. We report the scores for the 10-fold cross-validation scenario.

**RQ1** Overall, we are able to detect differences between translated and non-translated fiction in Russian. Combined features and each feature set individually performed better than the chance-level baseline, except collocational features for translated Spanish, which were on par with the dummy classifier. Remarkably, structural features (45 UD in Table 1) performed better than lexical features (11 n-gram and 12 collogram in Table 1). One explanation could be the size of the corpus underlying the language models, which does not provide enough evidence for the frequency of items in non-translated Russian. Note that our lexical features do not directly rely on the frequencies of individual items. Instead, they estimate the ratios of high- and low-frequency items in translated and non-translated subcorpora of text chunks. Combining two lexical feature sets brings the performance of the classifiers to the area of 70-81% accuracy, with the exception of translations from French (66%). It seems that n-grams and collograms complement each other in capturing lexical distinctions from the reference corpus. It is not surprising, given that frequencies of OOV n-grams and collograms are commonly picked among the top five predictors in each feature set by most of the classifiers. The combined features accumulate the effects of the individual feature sets and are harder to interpret. They can be used to generally demonstrate the extent to which translations are distinguishable from non-translations. In our experiment, the accuracies of translations from any SL on all features were in the range from 82% to 91%.

To explore the impact of the individual features on the classification outcome, we employed Recursive Feature Elimination (RFE) algorithm based on Support Vector Regressor (SVR), which selected a unique combination of N features. RFE selects features that return the best classification results. There were only two features shared in RFE-based selections by all classifiers, if N=30: the ratio of high frequency trigrams and the frequency of modal predicates (defined as the cumulative frequency of lemma мочь, lemma следовать with a dependent infinitive, three modal adverbs (можно, нельзя, надо) and 11 adjectives in the short form (e.g. должен, способный, возможный)). However, the frequency analysis for these features shows no significant differences in their values in many test subcorpora. Even if the difference was significant at p < 0.05,
the effect size did not exceed Cohen’s d of 0.2.

RQ2 Interestingly, the results from the binary classifications for lexical features, unlike UD features, are more volatile with regard to the assumption that translations from more distant languages are more predictable. On structural features, this statement holds with the bold exceptions of Finnish and Bulgarian which returned too low or too high classification results for 84-91% and 79-83% brackets for distant and close languages, respectively. However, on either of the lexical feature sets, there is hardly any consistency to be seen.

In feature selection, there were no features that cut across all language pairs among top N=10 and N=20 UD features. However, typologically similar languages shared up to five features. For example, the selections for English, German and Swedish included ‘ccomp’, ‘mquantif’, ‘xcomp’, ‘nnargs’, ‘ppron’. We interpret this finding as evidence of the SL traces in translations, leaving a more in-depth analysis of these features for future work.

To find out whether the SL signal in each language pair is strong enough to make the respective translations stand out of the bulk of other translations and non-translations, we applied a multiclass classification on the entire feature set. In this experiment, each of the 12 subcorpora (Russian reference and 11 source-language subcorpora) was classified against the rest of the subcorpora. We achieved the overall accuracy of 53% (F-measure of 0.53), which was well above the random baseline of 6%. The darker areas in the confusion matrix (Figure 2 in Appendix) shows that translations from same-family SLs were confused more often between themselves than with translations from distant languages. For instance, Belarusian is often confused with Ukrainian, but never with English and Spanish. In a similar manner, Russian non-translations were more likely to be misclassified as translations from Ukrainian, Belarusian or Latvian than from English or Swedish.

The results for individual languages also confirm the hypothesis that SL footprints specific for each language group are discernible in translations. Particularly, distant SLs generate translations that are more distinguishable as such than translations from closer languages. Better results (darker blue squares in Figure 2) are achieved for more distant languages, e.g. French, English and Swedish (F-measure of 0.62, 0.58 and 0.58, respectively). At the same time, the results are worse for typologically closer languages, such as Bulgarian and Latvian (both with an F-measure of 0.46). Against our expectations, the scores for the closest languages, such as Ukrainian and Belarusian, are not the lowest (0.50 and 0.54, respectively). This may be explained by fewer instances that we have for the lowest-scoring Bulgarian and Latvian.

5 Conclusion

We analysed translationese in literary texts, exploring differences between fiction originally authored in Russian and fiction translated into Russian from 11 languages. We found out that overall, we can automatically predict translations in this register with the accuracy well above the chance level. Besides, we compared performance of classifiers for translations from various SLs. We expected that translations from more distant SLs would return higher results in a series of binary classifications and would be easier to recognise in a multiclass setting than translations from typologically closer languages. This was confirmed for the subset of structural features, but not for lexical features, which returned mixed results. Possible explanations for the opposite tendencies observed in the data when using lexical features could be deficiency of the reference model and variation in literature style or literary translation tradition.

At the same time, we also understand the limitations of our data selection and research design. For example, experiments demonstrate that average sentence length might have been a better indicator of genre comparability than the length of a literary piece. The reference model for collocational features should probably be trained on a larger corpus to ensure greater coverage of lexical items. We leave detailed statistical analysis of the best-performing features for each language group and case studies based on parallel concordances for future work.

References


Marco Baroni and Silvia Bernardini. 2006. A new


A Appendix

<table>
<thead>
<tr>
<th>type</th>
<th>number</th>
<th>list of features</th>
</tr>
</thead>
<tbody>
<tr>
<td>morphological forms</td>
<td>8</td>
<td>past tense, passive voice form, finite and two non-finite forms of verb (infinitive and all participles), deverbal nouns, superlative and comparative degrees of comparison</td>
</tr>
<tr>
<td>word classes</td>
<td>7</td>
<td>personal, indefinite, possessive and demonstrative pronouns, adverbial quantifiers, coordinate and subordinate conjunctions</td>
</tr>
<tr>
<td>discourse markers</td>
<td>6</td>
<td>contrastive, additive, causative-consecutive, temporal-sequential connectives, epistemic stance markers and ‘but’ as a separate feature.</td>
</tr>
<tr>
<td>types of clauses</td>
<td>7</td>
<td>relative clauses, pied-piped subtype of relative clauses, coreferential constructions, adverbal clauses introduced by a pronominal adverb, adjectival clause, clausal complement, a predicative or clausal complement without its own subject</td>
</tr>
<tr>
<td>syntactic features</td>
<td>8</td>
<td>words in attributive function, modal predicates, auxiliary, passive voice auxiliary, copula verbs, nouns or proper names used in the functions of core verbal argument and as subject of a passive transformation, asyndeton</td>
</tr>
<tr>
<td>general text measures</td>
<td>9</td>
<td>lexical type-to-token ratio and lexical density (based on disambiguated content types), mean hierarchical and mean dependency distances, number of simple sentences, negative sentences, interrogative sentences, number of clauses per sentence, sentence length</td>
</tr>
</tbody>
</table>

Table 2: UD features: features extracted from the UD-annotated documents.

Figure 2: Confusion metrics for multiclass classification (the colour captures the number of predicted datapoints).
Corpus Creation and Language Identification in Low-Resource Code-Mixed Telugu-English Text

Kusampudi Siva Subrahamanyam Varma, Anudeep Chaluvadi, Radhika Mamidi
Language Technologies Research Centre, IIIT Hyderabad, Telangana, India
siva.subrahamanyam@research.iiit.ac.in
anudeep.chaluvadi@students.iiit.ac.in
radhika.mamidi@iiit.ac.in

Abstract

Code-Mixing (CM) is a common phenomenon in multilingual societies. CM plays a significant role in technology and medical fields where terminologies in the native language are not available or known. Language Identification (LID) of the CM data will help solve NLP tasks such as Spell Checking, Named Entity Recognition, Parts-Of-Speech tagging, and Semantic Parsing. In the current era of machine learning, a common problem to the above-mentioned tasks is the availability of Learning data to train models. In this paper, we introduce two Telugu-English CM manually annotated datasets (Twitter dataset and Blog dataset). The Twitter dataset contains more romanization variability and misspelled words than the blog dataset. We compare across various classification models and perform extensive bench-marking using both Classical and Deep Learning Models for LID compared to existing models. We propose two architectures for language classification (Telugu and English) in CM data: (1) Word Level Classification (2) Sentence Level word-by-word Classification and compare these approaches presenting two strong baselines for LID on these datasets.

1 Introduction

Language is one of the significant aspects which makes humans different from other species. It is not a fixed entity, and it has evolved with time and will continue to do so. As a part of such an evolutionary process, we are at the stage of Code-Mixing where people communicate by mixing linguistic units such as phrases, words, and morphemes of one language embedded within an utterance of another language (Sankoff and Poplack, 1981), (Poplack, 1980).

India has 22 officially recognized languages and many dialects. For a land with such linguistic diversity, bilingualism and multilingualism is a prevalent trait. Telugu belongs to the family of Dravidian languages. It is primarily spoken in Southern India and is also the third most spoken language in India. Telugu is the mother language of a large population native of Andhra Pradesh and Telangana states. English is a primary mode of teaching for most of the population across the globe. With such an influence of a language, people tend to use a mix of both English and their native language in an informal conversation (both speech and text).

CM is classified into two types: Intra-Sentential and Inter-Sentential Code-Mixing (Zirker, 2007). Intra-Sentential Code Mixing refers to the use of multiple languages in a single sentence. Inter-Sentential Code-Mixing is when the language switching is done at the end of the sentence. In this paper, we will focus on Intra-Sentential Code Mixing.

In a multilingual setting, most of the conversations that happen informally are CM. CM is also used extensively in Social Media platforms, blogs, and forums as posts, chats, and comments. The processing of CM text poses an exciting and challenging problem to the linguistic community. This is because of the added complexities in the traditional processing tasks such as Spell Checking, Named Entity Recognition (NER), Parts-Of-Speech (POS), Natural Language Generation (NLG) and Machine Translation (MT) due to the unavailability of prior information about the language at any point of time. Intra-Sentential LID is the task of identifying the languages

1 Eighth Schedule to the Constitution of India
of each word. Researchers have made significant progress in the LID module in the automated processing of CM text. However, in low-resourced agglutinative languages like Telugu, LID is still a challenging task (Parupalli et al., 2018).

In this paper, we introduce two datasets and propose various pipelines to tackle Intra-Sentential language identification problem in CM data using deep learning models. This example sentence illustrates the CM addressed in this paper:

**Example:** Elen/NE everyday/EN school/EN ki/TE buslo/TE velthundi/TE ./UNIV (Translation: Elen goes to school by bus everyday)

The words followed by /NE, /TE, /EN, and /UNIV correspond to Named Entity, Telugu, English, and Universal tags. In the above example, some words exhibit morpheme level Code-Mixing, like in “buslo” : “bus” (English word) + “lo” (plural morpheme in Telugu). We also consider the clitiques like “supere”: “super” (English root word) + “e” (clitique) as code mixed.

The **key contributions** of this paper are the following:

- We open-sourced\(^2\) two datasets of low-resourced CM English-Telugu data from popular global social media sites like Twitter and local blogging sites like Chaibisket.com, and Wirally.com.
- We propose extensive benchmarking with both Classical and Deep Learning Models for LID in CM data and infer that BiLSTM + CRF, BiLSTM + LSTM have higher classification metrics (overall and per-class) as compared with other models.
- We analyze the impact of contextual information of the word in a sentence for this task.

The rest of this paper is organised into 6 sections. In section 2, we discuss the related work. Section 3 elaborates on challenges while working with CM data, followed by the dataset and its annotation in section 4. Section 5 describes the approaches for LID in CM data. Section 6 reports the results of the proposed approaches. Finally, in section 7, we discuss the conclusions and future work.

## 2 Related Work

A significant amount of work has been done recently in the field of code-mixed data, especially in the area of LID. Kachru (1978) discussed the syntax and structure of multilingual language organization and the role of language dependence in linguistic convergence of CM from an Indian perspective.

To create a word-level language identifier, King and Abney (2013) used weakly semi-supervised methods. According to Noor Al-Qaysi (2017), code-switching is very popular on social networking sites such as Facebook, Twitter, and WhatsApp and 86.40 percent of students use code-switching on social networks, while 81 percent of educators do so.

Yogarshi Vyas and Choudhury (2014) used logistic regression and a module that calculates code-switching probability. Das and Gamb (2014) merged two classifiers into an ensemble model for Hindi-English CM LID using multiple features such as word sense, dictionary, n-gram, and edit distance. The first classifier takes changed edit distance, word frequency, and character n-grams as features. The first classifier’s output and the POS tag of the neighboring words are given as features to the second classifier to predict the final label.

Sharma (2007) used the shallow parsing pipeline to perform successful text analysis in Hindi-English CM social media data. For the task of LID, most of the experiments depend on dictionaries, supervised classification models, and Markov models.

The basic distinguishing features such as specific character combinations, repeated or unique words, diacritics, or typical n-grams are used in the simplest LID methods (Dunning, 1994; Clive Souter and Johnson, 1994; Ciprian-Octavian Truic˘a and AlexandruBoicea, 2015).

Some LID methods model sequences of words, characters, or bytes as model complexity increases. Some approaches concentrate on modeling the frequency of n-grams, such as character n-gram frequency (Bashir El-haj Ahmed and C.Tappert, 2004; Clive Souter and Johnson, 1994). These methods outper-
form techniques that rely on one-of-a-kind terms.


In (Gundapu and Mamidi, 2018), efforts have been made to propose an accurate model to address the problem of classification in CM data. However, the recent advancement in deep learning models have led to better performance in various NLP tasks. We thus leverage these deep models for classification involving low resource language (Telugu) in CM data with the help of larger corpus which have been sourced from Twitter and other popular blogs.

3 Challenges Observed in CM Data

1. **Gathering Data:** Data collection is the primary and the most crucial step while dealing with the problem with any neural-network-based approaches (Roh et al., 2021). There is a huge challenge for collecting CM data due to fewer resources and its informal nature of use in limited places. Datasets for low-resource languages like Telugu are challenging to find, making it difficult to build supervised models.

2. **Misspelled Words:** Since most of the data for CM comes from informal resources like social media posts, casual blogs, some of them are misspelled (refer Table 1). This is a significant challenge while building spell agnostic models.

3. **Romanization Variability:** One other defiance/challenge in CM data apart from spell checking is the variability in romanization output. For example, the Telugu word ‘enduku’ (Meaning: why) can either be written as ‘endhuku’ or ‘nduku’ (For more examples, refer Table 1).

4. **Feature Extraction:** Due to misspelled words and romanization variability, popular feature extraction methods like Word2Vec cannot be used due to high variations of the same word. We, therefore, used low-level features (explained in 5.1), which we found to be working well for this task.

5. **Morpheme-level CM:** As explained in section 1 example, Handling morpheme-level CM adds more complexity to the problem as the word is a combination of two words from two different languages.

In the next section, we explain the procedure used to create LID dataset for CM English-Telugu data.

4 Dataset

We created two different types of code-mixed datasets sourced from Twitter3 and popular blogs4 using Code-mixed language.

As shown in table 1, the Twitter dataset has significant variability in style, whereas the second dataset consists of sentences from articles written by professionals, hence have minimal variability. Table 1 shows the variability in styles of writing the same word in different ways across both the datasets.

For the Twitter dataset, we manually identified 40 user accounts who tweet with CM; these user accounts often tweet on different aspects such as Movies, Politics, and Sports. We used the Twitter API to get 1000 tweets of each user. For the blog dataset, we scrapped chaibisket.com and wirally.com websites which provide high-quality code-mixed content. We manually picked 350 articles containing code-mixed data for this dataset. The pre-processing of the data involved the following steps:

1. Converting tweets into sentences
2. Removing sentences containing Dravidian characters
3. Removing sentences that contain only English words and only Telugu words
4. Removing sentences with less than five words
5. Removing URLs and other similar tokens
6. Tokenizing emojis and hashtags

---

3Webpage: https://twitter.com/
4Webpage: https://chaibisket.com/
5Webpage: https://wirally.com/
Table 1: Table explaining the romanization and spelling variability in both the datasets

<table>
<thead>
<tr>
<th>Source Word</th>
<th>Blog Dataset</th>
<th>Twitter Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>enduku</td>
<td>enduku, endhuku, nduku</td>
<td></td>
</tr>
<tr>
<td>chepu</td>
<td>cheppu, chpu, chepu</td>
<td></td>
</tr>
<tr>
<td>meeku</td>
<td>meeku, meku, miku</td>
<td></td>
</tr>
<tr>
<td>hyderabad</td>
<td>hyderabad, hyd, hydbad</td>
<td></td>
</tr>
<tr>
<td>correct</td>
<td>correct, crct</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Statistics on words in Datasets

<table>
<thead>
<tr>
<th>Label</th>
<th>Blog</th>
<th>Twitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>50891</td>
<td>70751</td>
</tr>
<tr>
<td>Telugu</td>
<td>76003</td>
<td>68762</td>
</tr>
<tr>
<td>Named Entities</td>
<td>6146</td>
<td>36226</td>
</tr>
<tr>
<td>Universal</td>
<td>2608</td>
<td>36281</td>
</tr>
<tr>
<td>Total words</td>
<td>135648</td>
<td>212020</td>
</tr>
<tr>
<td>Avg Sentence length</td>
<td>14</td>
<td>8.6</td>
</tr>
<tr>
<td>Total Sentences</td>
<td>9657</td>
<td>24404</td>
</tr>
</tbody>
</table>

Table 3: Cohen-Kappa Score for Inter-Annotator Agreement

<table>
<thead>
<tr>
<th>Label</th>
<th>Blog</th>
<th>Twitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>0.95</td>
<td>0.96</td>
</tr>
<tr>
<td>Telugu</td>
<td>0.93</td>
<td>0.94</td>
</tr>
<tr>
<td>Named Entities</td>
<td>0.94</td>
<td>0.97</td>
</tr>
<tr>
<td>Universal</td>
<td>0.98</td>
<td>0.97</td>
</tr>
</tbody>
</table>

After preprocessing the data, it is then manually annotated into four classes, i.e., Telugu (TE), English (EN), Named Entities (NE), and Universal (Univ). Named Entities include names of people, organizations, and locations. Universal includes punctuation marks, acronyms, emojis, hashtags, and numbers.

4.1 Inter Annotator Agreement

Three people proficient in Telugu dialects and English language and having a linguistic background have annotated the dataset. We calculated Inter Annotator Agreement score using Cohen’s kappa score (Cohen, 1960) in order to assess the quality of both datasets. Table 3 reports the Inter Annotator Agreement scores.

5 Methodology

In this work, we consider LID in CM data as a classification problem where we label each word to its corresponding language class. With the advent of deep learning, many popular tasks like Named-entity recognition, Parts-of-speech identification have shown promising results (Singh et al., 2018), (Meftah and Semmar, 2018). In section 5.2 we explain an extensive set of deep learning models for LID in CM data. We show the quantitative results of LID in section 6. We provide the following architectures based on the type of input the model is supplied with to solve this problem:

1. Word Level Classification (WLC)(fig. 1): Given a word, we classify it into one of the four classes. This approach does not take advantage of the contextual information of the given word in the sentence. The example given below illustrates the input and output from this approach.
   **Input:** bagundi (Translation: good)
   **Output:** TE

2. Sentence Level word-by-word Classification (SLC)(fig. 2): Given a sentence, we predict each word’s label in the sentence. This approach utilizes the contextual information of the word in the sentence and predicts the output of each word. Below example illustrates the input and output from this approach.
   **Input:** Sai class ki velli book theesadu . (Translation: Sai went to class and opened a book)
   **Output:** Sai/NE class/EN ki/TE velli/TE book/EN theesadu/TE ./UNIV

In the following subsections, we explain our approach as a two-fold process: (1) Feature representation (2) Model Training.

5.1 Feature Representation

Feature representation plays a crucial role in the training of a deep learning model and increasing its efficacy. The following subsection introduces two types of feature representa-
Character Level Encoding (CLE)
In this approach, each character of the word is made into a one-hot encoding vector.

\[
FCLE : [c1 \times m, c2 \times m, ..., cn \times m] \times m
\]

Where \( c1, c2, ..., cn \) are one-hot encodings of the characters of a word of length \( n \).

Word Level Encoding (WLE)
In this approach, each word in a sentence is encoded as a vector having the following features:

- **Character N-Grams**: Character N-Grams of the word.
- **TF-IDF**: Term-Frequency and Inverse-Document Frequency of the N-Grams feature vector.
- **Hand-picked features**: The below high level features are chosen to capture semantics of various types of words such as Acronyms, numbers and punctuation.
  - Count of special characters
  - All capital letters
  - Starts with capital letter
  - Number of digits
  - Length of the word

Deep Learning Models
Deep learning models have been successful in understanding the semantic representations and learning complex tasks in Natural Language Processing (NLP). This section puts forward the various deep learning models we have implemented to solve LID in English-Telugu CM data. To provide an extensive benchmark, we also compare the above models to classical models such as Naive Bayes, SVM, Logistic Regression, and CRF in Table 6.

We divide our deep learning models into two categories. The first set of models namely: LSTM, CNN, RNN, MLP Network are used in our first architecture - WLC (fig 1) and the second set of models namely: BiLSTM + CRF, BiLSTM + LSTM are used in our second architecture - SLC (fig 2).

Word Level Classification (WLC)
In this method, we take each word and extract the features with CLE. The sequence of vectors from CLE is given to a Word Level Classifier, which then classifies the given word into four classes, viz. EN, TE, NE and UNIV. The following models use the pipeline illustrated in fig 1.

- **CNN**: CNNs are mostly used for Images,
but we have seen that CNNs perform on text as well in few applications. For the features, One-Hot encodings of each word are concatenated to get a 2-Dimensional grid, on which the CNNs are applied. We used two Convolutional Layers, one with 32 filters and the other 16 filters.

- **RNN, LSTM**: The major problems faced by CNNs are handling sequential data, considering only current input, and lack of memorization of previous inputs. All these shortcomings are better handled by RNNs. A Recurrent Neural Network works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer. Long Short-Term Memory (LSTMs), on the other hand, is a type of RNNs which prevent the vanishing gradient problem often found in RNNs.

The above-explained models (CNNs, RNNs, LSTMs) utilize positional information of characters either through a kernel convolution (CNN) or through hidden layer propagation (RNNs, LSTM). Thus, we also tried with an MLP model which lacks the above positional knowledge using the feature representation technique explained in 5.1.2.

**MLP**: A Multi Layer Perceptron (MLP) is a supervised feed forward neural network, which consists of an input, an output layer and few hidden layers. Each layer consists of a set of neurons that receives input from the previous layer and sends output to neurons in the next layer based on activation function. We added three layers of Dense Networks with 256, 64, and 32 neurons for each layer, respectively. Figure 3 shows the pipeline for LID using MLP.

### 5.2.2 Sentence Level Word-by-word Classification (SLC)

In this method, we input an entire sentence and extract features with WLE and classify the output of each word with deep learning models, namely BiLSTM + CRF and BiLSTM + LSTM.

- **BiLSTM + CRF**: Bi-directional LSTM was proposed by (Schuster and Paliwal, 1997), is a variant of LSTM which allows data to flow forward as well as backward in time. BiLSTM + CRF improves the performance by giving more context of previous and next occurring data to the model. BiLSTM + CRF have been known to work well in sequence labeling tasks (Poostchi et al., 2018), and hence we have used this model to carry out experimentation on our proposed datasets. Figure 4 illustrates the pipeline for this model.

- **BiLSTM + LSTM**: Each word in a sentence is passed to WLE for feature extraction and then passed to the BiLSTM layer. We then use the hidden outputs of the BiLSTM layer as inputs to the LSTM layer to make the final prediction. Figure 5 illustrates the pipeline for this model.

### 6 Results

We have put forward an extensive set of deep learning models to tackle LID in English-
Table 4: Per-Class Precision and Recall metrics on test data of Blog Dataset

<table>
<thead>
<tr>
<th>Classifier</th>
<th>TE Precision</th>
<th>TE Recall</th>
<th>EN Precision</th>
<th>EN Recall</th>
<th>NE Precision</th>
<th>NE Recall</th>
<th>UNIV Precision</th>
<th>UNIV Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>98.84</td>
<td>97.84</td>
<td>97.88</td>
<td>98.72</td>
<td>81.42</td>
<td>85.27</td>
<td>94.11</td>
<td>96.07</td>
</tr>
<tr>
<td>CNN</td>
<td>98.06</td>
<td>97.71</td>
<td>97.21</td>
<td>97.80</td>
<td>83.15</td>
<td>84.11</td>
<td>96.23</td>
<td>92.40</td>
</tr>
<tr>
<td>LSTM</td>
<td>98.73</td>
<td>98.37</td>
<td>98.07</td>
<td>98.54</td>
<td>85.62</td>
<td>86.32</td>
<td>95.56</td>
<td>95.13</td>
</tr>
<tr>
<td>RNN</td>
<td>92.22</td>
<td>98.27</td>
<td>91.00</td>
<td>95.89</td>
<td>0</td>
<td>0</td>
<td>96.65</td>
<td>58.82</td>
</tr>
<tr>
<td>BiLSTM + CRF</td>
<td>98.81</td>
<td><strong>99.27</strong></td>
<td>98.41</td>
<td>97.95</td>
<td>88.24</td>
<td>85.34</td>
<td>98.13</td>
<td>91.56</td>
</tr>
<tr>
<td>BiLSTM + LSTM</td>
<td><strong>99.04</strong></td>
<td>99.17</td>
<td>98.21</td>
<td>98.68</td>
<td>89.98</td>
<td><strong>86.40</strong></td>
<td><strong>99.37</strong></td>
<td>92.73</td>
</tr>
</tbody>
</table>

Table 5: Per-Class Precision and Recall metrics on test data of Twitter Dataset

<table>
<thead>
<tr>
<th>Classifier</th>
<th>TE Precision</th>
<th>TE Recall</th>
<th>EN Precision</th>
<th>EN Recall</th>
<th>NE Precision</th>
<th>NE Recall</th>
<th>UNIV Precision</th>
<th>UNIV Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>97.69</td>
<td>97.77</td>
<td>97.38</td>
<td>98.37</td>
<td>96.56</td>
<td>94.21</td>
<td>96.39</td>
<td>96.74</td>
</tr>
<tr>
<td>CNN</td>
<td>96.76</td>
<td>96.29</td>
<td>96.42</td>
<td>96.61</td>
<td>95.09</td>
<td>94.09</td>
<td>95.34</td>
<td>96.90</td>
</tr>
<tr>
<td>LSTM</td>
<td>98.25</td>
<td>97.36</td>
<td>98.03</td>
<td>98.69</td>
<td>97.13</td>
<td>96.63</td>
<td>96.56</td>
<td>97.46</td>
</tr>
<tr>
<td>RNN</td>
<td>89.03</td>
<td>95.08</td>
<td>92.34</td>
<td>94.61</td>
<td>96.14</td>
<td>84.52</td>
<td>97.39</td>
<td>97.70</td>
</tr>
<tr>
<td>BiLSTM + CRF</td>
<td><strong>99.52</strong></td>
<td><strong>99.35</strong></td>
<td><strong>99.14</strong></td>
<td><strong>99.17</strong></td>
<td><strong>99.21</strong></td>
<td><strong>99.53</strong></td>
<td><strong>99.35</strong></td>
<td><strong>99.00</strong></td>
</tr>
<tr>
<td>BiLSTM + LSTM</td>
<td>98.03</td>
<td>98.80</td>
<td>98.34</td>
<td>98.57</td>
<td>97.55</td>
<td>97.05</td>
<td>99.14</td>
<td>97.76</td>
</tr>
</tbody>
</table>

Figure 5: BiLSTM + LSTM model.

Telugu CM data in our present work. To validate these models, we show quantitative results on both the proposed datasets, namely Twitter and Blogs datasets (explained in section 4). We also compare with the existing classical Machine Learning models (refer Table 6). We show Precision and Recall metrics for each class in table 4, 5. It is observed that RNN had faced the problem of vanishing gradients, thus precision and recall of RNN for NE is zero. The BiLSTMs property of propagating contextual information in both directions helps it to have the edge over WLC models like MLP, CNN, and RNN models. From table 4, 5 we see that the BiLSTM + LSTM, BiLSTM + CRF models outperform the other

Deep learning models in this task and achieves an improvement in accuracy over the classical models (Baseline: CRF) of around 2.38%, 2.09% on the Blog and Twitter data-set respectively. It can also be noted that, though our primary task is to identify Telugu and English words in CM Data, from (from 4, 5) we observe that the precision and recall of the Named Entity class is on the higher side in

Table 6: Model testing accuracy of classical models and Deep Learning models on Blog and Twitter Datasets

<table>
<thead>
<tr>
<th>Model</th>
<th>Blog</th>
<th>Twitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
<td>88.26</td>
<td>86.43</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>94.18</td>
<td>93.59</td>
</tr>
<tr>
<td>SVM</td>
<td>90.67</td>
<td>86.85</td>
</tr>
<tr>
<td>CRF</td>
<td><strong>96.15</strong></td>
<td><strong>97.23</strong></td>
</tr>
<tr>
<td>MLP</td>
<td>97.54</td>
<td>97.17</td>
</tr>
<tr>
<td>CNN</td>
<td>96.98</td>
<td>96.11</td>
</tr>
<tr>
<td>LSTM</td>
<td><strong>97.78</strong></td>
<td><strong>97.70</strong></td>
</tr>
<tr>
<td>RNN</td>
<td>91.80</td>
<td>92.57</td>
</tr>
<tr>
<td>BiLSTM + CRF</td>
<td>98.35</td>
<td><strong>99.32</strong></td>
</tr>
<tr>
<td>BiLSTM + LSTM</td>
<td><strong>98.53</strong></td>
<td>98.24</td>
</tr>
</tbody>
</table>
SLC (BiLSTM) as compared to WLC models (MLP, CNN, LSTM, RNN). We also see that the accuracy of RNN falls as it suffers from vanishing gradients, which hampers the learning of long data sequences as observed in the Twitter dataset.

One of the few challenges that were encountered was the Romanization of Telugu words and social media acronyms and abbreviations. As explained in section 3, there is no standard way to transliterate the code mixed data, and thus Romanization variability leads to different spellings of the same word. For example, the romanization variability of a single word can be: “eppudu”, “epdu”, “epudu”, “yepudu” (Translation into English: “when”). Similarly, social media chat conversations/tweets using SMS language “you” can be written as “U”, “hello” as “helooo”, “What’s up” as “wassup” etc. All these examples pose a significant challenge while training the LID models in code mixed data.

7 Conclusion and Future Work

In this paper, we have put forward two Telugu-English CM manually annotated datasets which are an order of magnitude greater than the existing dataset, and proposed two architectures for language classification (Telugu and English) in CM data: (1) Word Level Classification (2) Sentence Level word-by-word Classification. We have conducted thorough experimentation and extensive benchmarking across various Deep Learning models and Classical Machine Learning models. We found out that BiLSTM + LSTM, BiLSTM + CRF performs the best among others. We also plan to make our data corpus consisting of low-resource Telugu and English languages generated from Twitter, and online blogs opensourced to encourage further experimentation and research. The LID models developed here can also be used in other NLP tasks like Named Entity Recognition (NER), Parts of speech (POS) tagging, and spell-check. The results of the current work are encouraging, and future work will be focused on using sequence level labeling SOTA models like attention in LID. We are also focused on developing spell-checking models which normalize the misspelled and romanization variability as a part of our future work.

References


Amitava Das and Bjorn Gamb. 2014. Identifying languages at the word level in code-mixed indian social media text.


Kelly Ann Hill Zirker. 2007. Intrasentential vs. intersentential code switching in early and late bilinguals.
Abstract

In a multilingual society, people communicate in more than one language, leading to Code-Mixed data. Sentimental analysis on Code-Mixed Telugu-English Text (CMTET) poses unique challenges. The unstructured nature of the Code-Mixed Data is due to the informal language, informal transliterations, and spelling errors. In this paper, we introduce an annotated dataset for Sentiment Analysis in CMTET. Also, we report an accuracy of 80.22% on this dataset using novel unsupervised data normalization with a Multilayer Perceptron (MLP) model. This proposed data normalization technique can be extended to any NLP task involving CMTET. Further, we report an increase of 2.53% accuracy due to this data normalization approach in our best model.

1 Introduction

In recent times, huge volumes of data are being generated on social media and online blogging. The usage of multiple languages in day-to-day conversations and the minimal linguistic restrictions on the online content lead to the generation of Code-Mixed data. Code-Mixing or Code-Switching is the usage of two or more languages in a single sentence or a conversation. The generated code-mixed data can be used to extract potential knowledge like emotion, news or information.

Sentimental analysis on Code-Mix social media data helps us to understand the underlying sentiment of the phrase or sentence, which can have many practical use cases in the real world. For example, it can be used to understand the sentiment of restaurant or movie reviews.

Code-Mixed Data can be easily extracted from various online platforms like social media and blogs using APIs and various web-scraping tools. In spite of having huge amounts of data, performing sentiment analysis on it is still challenging because of its unstructured and noisy nature (Arora and Kansal (2019), Gautam and Yadav (2014), Barik et al. (2019)). Hence, this requires robust preprocessing techniques to normalize this unstructured data.

In this paper, we have performed sentimental analysis on Code-Mixed Telugu-English Text (CMTET) with an unsupervised data normalization. We analyse and classify each sentence into three sentiments, positive, negative and neutral. To the best of our knowledge this is the first work to propose a framework to perform sentiment analysis on CMTET. The reason for choosing English as our secondary language for the Code-Mixed data is because we observed that most of the Telugu data available in social media is very often Code-Mixed with English.

The rest of the paper is organized as follows. In section 2, we discuss the existing Sentiment Analysis approaches in other Code-Mixed languages. Section 3 introduces the dataset and describes the methodology involved in preprocessing and annotating this dataset. In section 4, we discuss in detail various challenges faced in performing sentiment analysis in CMTET. In section 5, we discuss about the proposed data normalization technique for CMTET. In section 6, we explain the process of feature extraction and sentiment classification. In Section 7, we show the results of the proposed methods and make a comparative study on the accuracy of all those models. In section 8, we discuss the problems faced by the models in predicting the sentiment. Section 9 concludes the paper with the summary and scope for future work.
2 Related Work

For the task of Sentiment Analysis on Code-Mixed Hindi-English text, Sharma et al. (2015) used a lexicong-based approach on the FIRE 2013 (Roy et al., 2013) and FIRE 2014 datasets. The dataset consisted of the user comments from public Facebook pages of two of the most popular celebrities. Joshi et al. (2016) introduced a new dataset and used subword-LSTM to address the noisy nature of the text and reported an improvement of 18% on the baseline. Choudhary et al. (2018) proposed clustering of Code-Mixed word variations with skip-gram vectors based on similarity. They also used contrastive learning and projected the Code-Mixed sentences into a common sentiment space using shared parameters. Sukhpreet Kaur (2021) used attention based models with word-level, sub-word level and character-level representations of the sentences, and reported that Bi-LSTM performed the best.

Chakravarthi et al. (2020) had presented an annotated code-mixed corpus in Malayalam-English language of Youtube comments for the task of Sentiment Analysis. Kalaivani and Thenmozhi (2020) performed Sentiment Analysis in Code-Mixed Malayalam-English and Tamil-English text using AWD-LSTM(Merity et al., 2017) and reported a weighted F1-Score of 0.6 on both the datasets. Mishra et al. (2018) proposed an ensemble model to perform sentiment analysis in Hindi - English and Bengali - Hindi - English corpus. The ensemble model is built on linear SVM, logistic regression and random forest based on a voting classifier.

Sabri et al. (2021) created a Persian-English code mixed data corpus from tweets. They also proposed a Bi-LSTM based ensemble model which uses BERT embeddings and translation models to learn sentiment of tweets. Yadav and Chakraborty (2020) proposed a zero-shot approach to solve Sentiment Analysis task on Code-Mixed Spanish-English text. They used multilingual and crosslingual embeddings to transfer knowledge from monolingual text to Code-Mixed text. They reported an increase of 3% in accuracy over the previous state-of-the-art.

In the task of Sentiment Analysis on Social Media text, there is a lot of research on using different models to address the problem of its noisy nature. Our work differs in this context, where, we propose an unsupervised approach for normalizing CMTET. And to the best of our knowledge, this is the first work of Sentiment Analysis in CMTET.

3 Dataset

In this paper, we introduce a new dataset for sentiment analysis in CMTET. The methodology consists of three main phases: data collection, data cleaning, and data annotation. The below sections describe each phase in detail.

3.1 Data Collection

We have identified a few Twitter users and regional movie review Youtube videos. We have observed that these Twitter users tweet in CMTET on different aspects such as Movies and Sports. Also, the identified Youtube videos contain user comments expressing their sentiment of the movie or the video in CMTET. We used Twitter public streaming API\(^1\) and Youtube Comments API\(^2\) to collect this data.

3.2 Data Cleaning

We removed irrelevant text such as URLs and markup text, using regex-based\(^3\) pattern matching to ensure basic data quality. Then, with the help of NLTK Tokenizer\(^4\), we tokenized each sentence into words. We then removed sentences containing less than five words, as we observed that these sentences are noisy and hardly contain any information.

3.3 Data Annotation

We adopted the three-class classification for sentiment of the sentences, i.e., positive, negative, and neutral (Koppel and Schler, 2006). The objective of this phase is to annotate each sentence. We also annotated this data at word-level with their language using the language tags, i.e., English (EN), Telugu (TE), Named Entities (NE), and Universal(UNIV).

---

\(^1\)https://developer.twitter.com/en/docs/twitter-api
\(^2\)https://developers.google.com/youtube/v3/docs/comments/list
\(^3\)https://en.wikipedia.org/wiki/Regular_expression
\(^4\)https://www.nltk.org/api/nltk.tokenize.html
After the word-level annotation, we removed the sentences having only English or only Telugu words.

The annotation was carried out by 5 Telugu native speakers who are also proficient in English. We developed an efficient annotation process with the help of Telegram Bot API\(^5\), where the annotator can annotate from their Telegram App with a single tap on their device. Figure 1 shows a screenshot of the user interface of the Telegram Bot made for this task.

In most of the earlier works, the annotation task has been done by developing an application, which is usually web-based (Aprosio et al. (2020), (Wadden et al., 2020)). Using this process would either have device compatibility or user-experience issues. There will be a significant increase in application development overhead to address these issues. On the other hand, using a Telegram Bot API offered excellent user experience, lesser maintenance, lesser development overhead, and its availability in most devices (both mobile and desktop). Our annotators gave us positive feedback for the tool, pointing out the flexibility offered to perform the annotation task in any environment.

Inter Annotator Agreement: We calculated the Inter Annotator Agreement score using Cohen’s Kappa score (Cohen, 1960) in order to assess the quality of the dataset. We

\[ \text{Sentiment: Positive} \]

At the end of data annotation we got a total of 19,857 sentences of which 7,925 are Positive, 7,713 are negative and 4,219 are neutral sentences. The data is open-sourced\(^6\) to encourage further research.

Sample Data:

Sentence: super/EN bro/EN nuvvu/TE morn-ing/EN adiga/TE appude/TE review/EN icharu/TE tnq/EN soo/EN much/EN

Sentiment: Positive

4 Challenges in CMTET

The CMTET poses new challenges because of its unstructured nature due to the following phenomenons.

4.1 Informal Transliterations

Users in social media follow no standard when transliterating Telugu from Dravidian to Roman text. Hence many transliteration variants are observed in the CMTET. Below, we discuss the various transliterations observed.

4.1.1 Long Vowels

People tend to transliterate long vowels in many ways. For example, the word తినాన్వా (winnA vA) is transliterated into tinnaaavaa

\[ \text{https://github.com/ksubbu199/cmtet-sentiment} \]
(repeating the long vowel twice) or just tinnava (not indicating the long vowel at all) or tinnavaa, tinnaava (mixture of double and single vowels).

### 4.1.2 Double Consonants

Similar to long vowels, even double consonants are transliterated in many ways. For example, ཕྲན་མ། (winnA vA) is transliterated into tinnava, tinava or བྲན་ཕྲ (sariggA) into sariggaa, sarigaa.

### 4.1.3 Aspirated Consonants

Aspirated Consonants are the syllables which require burst of breath to pronounce. In Telugu ఖ, ఛ, ఘ, ఠ, ఝ, భ, ఠ are the aspirated consonants. In CMTET, we observed that these characters are transliterated in multiple ways. For example, ధర (xara) is transliterating into dhara or dara.

### 4.1.4 Homophones

Homophonic syllables are usually transliterated in multiple ways due to their nature of having same pronunciation with different spellings. For example, ఉనాన్యి (unnA yi): unnayi, unnai or ఎక్కడా (ekkada): ekkada, aekkada, akkada.

### 4.2 Informal Language

Some of the variations in the spellings are caused due to the lack of formal setting on social media. Following are some variations caused by the informal setting of social media.

#### 4.2.1 Elongation

To express certain sentiments like excitement, users stretch some words in an informal setting. For example: helloooo, niceeee, gooood, okayyy, bagundhiii, ichaavv.

#### 4.2.2 Shortening

Due to limited characters in Twitter, users tend to shorten words, yet capturing the word’s phonetics. For example, plz, grt, cref, ndku (నాడును). In Telugu, shortening is usually done by dropping vowels or using single characters for double consonants.

### 4.3 Spelling and Typing Errors

Ritter et al. (2010), in their modeling of Twitter conversations, found that posts were “often highly ungrammatical, and filled with spelling errors” and resorted to selecting clusters of spelling variations manually. In CMTET, we observed spelling errors in both English and Telugu.

From the above, we can understand that CMTET has high entropy in spellings and thus poses a lot of challenges. To resolve these challenges, we propose an unsupervised data normalization technique for CMTET.

### 5 Data Normalization

In this section, we propose an unsupervised data normalization technique for normalizing CMTET. The architecture for data normalization can be found in Fig 2. As a first step, we performed elongation normalization and then used the language tags in the dataset to normalize Telugu words and English words separately. We then spell-checked English words using a defined similarity metric with an English dictionary. For Telugu words, we performed a two-stage normalization to normalize transliteration and spelling errors. The below sections explain this architecture in detail.

#### 5.1 Elongation Normalization

To deal with the problem of Elongation (refer 4.2.1), we convert each character to lower-case, and then limit the repetitions of sequential characters to two. For example, hel loo to helloo, nicely to nicely, goooo to goood, bagundhi to bagundhii. Errors persisting after this step like hel loo and bagundhi are treated as spelling errors and are normalized in the next steps.

#### 5.2 Normalizing English Words

To address spelling and typing errors, we have used dictionary-based spell-check with Levenshtein Distance (Levenshtein, 1966) as a similarity score between two words. We used SymSpell\(^7\) to compute this efficiently.

#### 5.3 Normalizing Telugu Words

In Telugu, the objective is to cluster the vocabulary into groups capturing the transliteration variants and spelling errors of each word. We propose a two-stage normalization for this. The below sections will explain this method in detail.

---

\(^7\)https://github.com/wolfgarbe/SymSpell
Figure 2: Proposed pipeline to normalize transliteration variants and spelling errors in CMTET

Table 1: Captured transliteration variations with the proposed normalization method

<table>
<thead>
<tr>
<th>Standard Form</th>
<th>Meaning</th>
<th>Captured Variations</th>
</tr>
</thead>
<tbody>
<tr>
<td>తరువాత (waruvAwa)</td>
<td>After</td>
<td>tharuvatha, taruvata, tharvatha, tarvataa, taruvatha</td>
</tr>
<tr>
<td>వీడికి (vEdiki)</td>
<td>For him</td>
<td>veediki, vediki, vedki, vidiki</td>
</tr>
<tr>
<td>చెతత్ (cewwa)</td>
<td>The worst</td>
<td>chethha, chetha, chetha, cheta</td>
</tr>
<tr>
<td>అబదాధ్లు (abaxXAlu)</td>
<td>Lies</td>
<td>abaddhaalu, abbaddalu, abhadhulu, abadhal</td>
</tr>
<tr>
<td>ఎందుకు (enxuku)</td>
<td>Why</td>
<td>endhuku, endku, endhuku, enduk, endhuk, nduku</td>
</tr>
<tr>
<td>ఈరోజు (ErOju)</td>
<td>Today</td>
<td>eerooju, eroju eeroju, eorju, eorjuu</td>
</tr>
</tbody>
</table>

5.3.1 Normalizing Transliterations
In this stage, we aim to capture all the transliteration variations mentioned in Section 4.

- Limiting Character Repetition to One: This helps to address the issue of Long Vowels (refer 4.1.1) and Double Consonants (refer 4.1.2) transliterations. For example, tinnaavaa to tinava and sariigga to sariiga.

- Normalizing Aspirated Consonants: The transliterations of aspirated consonants in Telugu i.e., kh (.Sdk), chh (చే), gh (ఘే), th (ఠ), jh (ఝ) dh (డ) and bh (భే) are replaced with k, ch, g, t, j, d and b respectively. This will address the problem of Aspirated Consonant Transliterations (refer 4.1.3).

5.3.2 Clustering with PBLD
In this stage, we aim to normalize Homophones and spelling errors. As there is no standard dictionary for the transliterated Telugu text we aim to normalize the text by clustering them. We have experimented with Levenshtein Distance(LD)(Levenshtein, 1966) as a similarity score to cluster Telugu words. But, we observed a limitation that, LD treats all the characters equally leading to the clustering of wrong words. For example: According to LD, rasthaaru (రాసాత్రు:rAswAru) and vasthaaru (వసాత్రు:vaswAru) are unit distant. To address this issue, we propose a modified LD called Phonetic Based Levenshtein Distance (PBLD) with the following changes:

- Insertions and deletions are allowed only if they are vowels. This will address the issue of Shortening (refer 4.2.2) in Telugu.

- Characters can only be substituted with other characters if they have similar phonetics. This will address the variations in transliteration of Homophones (refer 4.1.4).

Table 2: Clustering error and Vocabulary Reduction with clustering transliteration variants with varying Edit Distance ($d$).

<table>
<thead>
<tr>
<th>$d$</th>
<th>Vocab. Reduction</th>
<th>Error %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LD</td>
<td>PBLD</td>
</tr>
<tr>
<td>1</td>
<td>32.4%</td>
<td>5.5%</td>
</tr>
<tr>
<td>2</td>
<td>57.8%</td>
<td>10.17%</td>
</tr>
<tr>
<td>3</td>
<td>70.65%</td>
<td>18.12%</td>
</tr>
</tbody>
</table>

762
### Table 1: Quantitative results across various Machine learning models

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameter</th>
<th>Without Data Normalization</th>
<th>With Data Normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>F1-Score</td>
</tr>
<tr>
<td>NB</td>
<td>Overall</td>
<td>77.33</td>
<td>67.25</td>
</tr>
<tr>
<td></td>
<td>Positive</td>
<td>67.05</td>
<td>89.58</td>
</tr>
<tr>
<td></td>
<td>Negative</td>
<td>81.87</td>
<td>82.33</td>
</tr>
<tr>
<td></td>
<td>Neutral</td>
<td>83.08</td>
<td>29.83</td>
</tr>
<tr>
<td>LR</td>
<td>Overall</td>
<td>74.73</td>
<td>74.27</td>
</tr>
<tr>
<td></td>
<td>Positive</td>
<td>78.22</td>
<td>80.92</td>
</tr>
<tr>
<td></td>
<td>Negative</td>
<td>82.53</td>
<td>82.46</td>
</tr>
<tr>
<td></td>
<td>Neutral</td>
<td>63.44</td>
<td>59.43</td>
</tr>
<tr>
<td>RF</td>
<td>Overall</td>
<td>73.75</td>
<td>73.01</td>
</tr>
<tr>
<td></td>
<td>Positive</td>
<td>78.55</td>
<td>76.80</td>
</tr>
<tr>
<td></td>
<td>Negative</td>
<td>77.67</td>
<td>83.67</td>
</tr>
<tr>
<td></td>
<td>Neutral</td>
<td>65.03</td>
<td>58.56</td>
</tr>
<tr>
<td>SVM</td>
<td>Overall</td>
<td>75.83</td>
<td>76.71</td>
</tr>
<tr>
<td></td>
<td>Positive</td>
<td>86.99</td>
<td>76.17</td>
</tr>
<tr>
<td></td>
<td>Negative</td>
<td>84.38</td>
<td>82.85</td>
</tr>
<tr>
<td></td>
<td>Neutral</td>
<td>56.14</td>
<td>71.11</td>
</tr>
</tbody>
</table>

### 5.3.3 Error Analysis

In this section, we perform an error analysis on normalization of Telugu words with LD and PBLD. Clustering of words leads to the reduction of Telugu vocabulary in the dataset. We randomly picked clusters having a total of 500 words and observed a significant difference between the two methods in terms of clustering error. These metrics are reported in Table 2.

### 6 Method

In this section, we focus on explaining our Sentiment analysis pipeline. Fig. 3 shows our end to end approach wherein we take raw data i.e. a sentence and perform data normalization (transliteration and spelling normalization) as explained in section 5. Once we have this normalized data we perform feature extraction using N-Grams and Term-Frequency and Inverse-Document Frequency (TF-IDF) (Chowdhury, 2010). These features are then passed to our sentiment classification models which outputs one of the labels namely positive, negative and neutral.

We experimented on Logistic Regression (LR), Naive Bayes (NB), Support Vector Machine (SVM), Random Forest (RF) and Multi Layer Perceptron (MLP) classification models to classify the final sentiment.

### 7 Experiments and Results

The motivation of the current work is achieve better Sentiment Analysis in CMTET using machine learning models. Majority of the existing approaches majorly focus on single language sentiment analysis (Zulkifli and Lee,
expected

Table 3: Error Analysis with MLP

<table>
<thead>
<tr>
<th>Text</th>
<th>Expected</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>vadu manchi director em kaadu, oka hit kuda leduu (He is not a good director, he don't even have a successful movie)</td>
<td>Negative</td>
<td>Positive</td>
</tr>
<tr>
<td>feel ayithae cheppu bro, hurt cheyali ani analedlu (Please let me know if you got hurt, I didn't mean to hurt you)</td>
<td>Neutral</td>
<td>Negative</td>
</tr>
<tr>
<td>climax maataram shawshank redemption la undhi bro &lt;3 (Bro, climax is like Shashawnk Redemption &lt;3)</td>
<td>Positive</td>
<td>Neutral</td>
</tr>
</tbody>
</table>

Table 4: Accuracy metrics of various ML models with two different approaches

<table>
<thead>
<tr>
<th>Models</th>
<th>Without Data Normalization</th>
<th>With Data Normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>74.06%</td>
<td>74.23%</td>
</tr>
<tr>
<td>LR</td>
<td>76.95%</td>
<td>78.76%</td>
</tr>
<tr>
<td>RF</td>
<td>76.31%</td>
<td>77.65%</td>
</tr>
<tr>
<td>SVM</td>
<td>75.59%</td>
<td>76.98%</td>
</tr>
<tr>
<td>MLP</td>
<td>77.69%</td>
<td>80.22%</td>
</tr>
</tbody>
</table>

9 Conclusion and Future Work

In this paper, we published a huge annotated dataset for sentiment analysis in CMTET to encourage further research. Also, we presented a pipeline for this task with a novel data normalization technique. For each model, we have shown quantitatively that the proposed data normalization improves the overall performance across various metrics (precision, recall and F1-score) (refer Table 4). To the best of our knowledge this is the first such method carrying out sentiment analysis on CMTET.

Although the current work addresses some of the most crucial challenges in sentiment analysis on CMTET it can further be extended to other languages. The proposed data normalization technique can also be leveraged in various other NLP tasks. We can further make the bench-marking more extensive by including other deep learning models like LSTM, RNNs and CNNs.

References


8https://www.imdb.com/title/tt0111161/
Alessio Palmero Aprosio, Stefano Menini, and Sara Tonelli. 2020. Creating a multimodal dataset of images and text to study abusive language.


Abstract

This paper deliberates on the process of building the first constituency-to-dependency conversion tool of Turkish\(^1\). The starting point of this work is a previous study in which 10,000 phrase structure trees were manually transformed into Turkish from the original Penn Treebank corpus. Within the scope of this project, these Turkish phrase structure trees were automatically converted into UD-style dependency structures, using both a rule-based algorithm and a machine learning algorithm specific to the requirements of the Turkish language. The results of both algorithms were compared and the machine learning approach proved to be more accurate than the rule-based algorithm. The output was revised by a team of linguists. The refined versions were taken as gold standard annotations for the evaluation of the algorithms. In addition to its contribution to the UD Project with a large dataset of 10,000 Turkish dependency trees, this project also fulfills the important gap of a Turkish constituency-to-dependency conversion tool, enabling the quick compilation of dependency corpora which can be used for the training of better dependency parsers.

1 Introduction

There are two types of annotated treebanks used in natural language processing (NLP) systems: constituency treebanks and dependency treebanks. They are used to represent syntactic relations, argument structures, and other hierarchical relations. Constituency treebanks display groups of phrases as trees and dependency treebanks marks head-dependent relations for each item. Today, dependency parsers are expected to adapt to new data from a variety of genres. The Universal Dependencies (UD) Project (Nivre et al., 2020) provides convenient corpora for these parsers. However, this kind of an adaptation requires a huge amount of data that is not suitable for manual annotation. Thus, the existent constituency treebank corpora are being automatically converted into dependency treebanks as the need for dependency treebanks increased in order to train dependency parsers (Marneffe et al., 2006; Johansson and Nugues, 2007; Choi and Palmer, 2012).

Constituency to dependency conversion requires a big corpus and a coherent annotation framework. UD helps to provide a framework for annotated treebanks to be used in different languages. There have been several attempts made to represent Turkish language in this project and to create a common framework in Turkish language treebank studies (Sulubacak et al., 2016; Oflazer et al., 2003; Cöltekin, 2015) (Kuzgun et al., 2020). However, Turkish language
has flexible word order as a consequence of its agglutinative morphology. Such features complicates the annotation scheme for Turkish and renders the language harder to parse while giving rise to problems such as non-projectivity. In their work, Türk et al. (2019) edited the IMST Treebank (Sulubacak et al., 2016) according to the UD framework considering the needs of the Turkish language but its corpus does not provide a parallel treebank in terms of a cross-linguistic attribution to the literature and even this refinement could not help the IMST Treebank to adapt the new versions of the UD framework. In spite of all the attempts and refinements which have been mentioned above, Turkish still lags behind compared to similar languages in the area. For instance, a dependency conversion tool is yet to be developed for Turkish while there are studies present for similar languages such as Hungarian (Simkó et al., 2014).

We have applied our conversion algorithms to the biggest Turkish Constituency Treebank which is the Turkish counterpart of the original Penn TreeBank corpus (Marcus et al., 2002), the first annotated treebank of English that set the gold standard for other treebanks. There have been several studies done for the parallel constituency treebank creation in Turkish. Yıldız et al. (2014) developed an automatic translation process to translate trees from the Penn Treebank. They created a tool that learned from the 5000 manually translated sentences and offered possible translations. Yıldız et al. (2015) fine tuned this work by deleting empty projections, and rearranging the two word constructions that were appearing in one node due to the cross-linguistic consequences. In their study, Bakay et al. (2019) translated sentences using a tree-based approach to deal with the long distance dependencies that are not present in the corresponding English sentences. These long distance dependencies occur as a result of the difference between English and Turkish in having a fixed word order and a flexible word order respectively.

The Turkish Penn Constituency Treebank used in this conversion study, however, is distinct from the previous parallel Turkish constituency treebank studies as the most recent one is translated and annotated manually. It consists of 10,000 annotated sentences\(^2\) translated from the original version of the Penn Treebank (Kara et al., 2020). In order to make the translations suitable for the Turkish language, a team of linguists deleted, changed some tags from the original Penn Treebank, and also they introduced new tags that are necessary for a better syntactic representation of the Turkish language. We preferred using The Turkish Penn Treebank in our conversion study as it offered an accurate and less complicated constituency treebank that is necessary for the demands of our conversion tool.

Our main objective in this study is to introduce the first Turkish constituency to dependency converter by using the parallel Turkish constituency treebank introduced above. Another contribution of this work to the literature comes from its cross-linguistic nature. Since most of the conversion studies used the Penn Treebank corpus (Mareeffe et al., 2006); (Johansson and Nugues, 2007); (Choi and Palmer, 2012), our choice of employing a translated corpus results in a Turkish-English parallel constituency to dependency conversion study.

All the previous converter studies employed a rule-based approach. Johansson and Nugues (2007) differs from these studies in that they included extra labels to include semantic information and to handle syntactic phenomena such as topicalization, clefting, gapping, and so on. They put the semantic annotations in the Penn Treebank in use to achieve a more semantically rich dependency treebank. However, as the labels were more complicated, the employment of semantic information decreased the parsing accuracy. The Stanford dependency approach, on the other hand, did not use these function labels that were manually annotated over the Penn Treebank corpus. In their study, Choi and Palmer (2012) combined the Stanford approach with the CoNLL dependency approach which uses different labels and relation rules. This combination allowed them to achieve better accuracy without eliminating the semantic information encoded in the Penn Treebank corpus.

In our study, the translated version of the Penn Treebank does not include these semantic labels, and therefore our conversion rules does not employ these semantic information as in the Stanford approach.

\(^2\)https://github.com/olcaytaner/TurkishAnnotatedTreeBank
<table>
<thead>
<tr>
<th>Hierarchy</th>
<th>Tag List</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VP, NOMP</td>
</tr>
<tr>
<td>2</td>
<td>S, NP, ADJP, ADVP</td>
</tr>
<tr>
<td>3</td>
<td>PP</td>
</tr>
<tr>
<td>4</td>
<td>DP, NUM</td>
</tr>
<tr>
<td>5</td>
<td>QP, NEG, CONJP, INTJ, WP</td>
</tr>
</tbody>
</table>

Table 1: Hierarchical head table.

2 Conversion Process

2.1 Turkish Penn Treebank as Input

Using the Penn Treebank as an input for converter algorithms is a very common procedure. So as not to break this tradition, we have also decided to use the Turkish version of Penn Treebank. As mentioned above, this widely-used constituency treebank was first adapted to Turkish by Yıldız et al. (2014). We found the latest version of the dataset to be simplistic and yet an adequate representation of the Turkish language, especially thanks to its being manually edited by a team of linguists. The tree in Figure 1 illustrates the final version of a sentence in the Turkish Penn Treebank corpus.

The input we used follows the main principles of the Penn Treebank annotation. However, there are differences in some aspects. In their study, Kara et al. (2020) excluded the bar level projections such as NONE, VBN, NNP, IN, NSBJ-1. They aimed for a more minimal approach to reduce the number of branches while keeping the necessary constituency information. In addition to this difference, this version has additional tags such as NEG, NOMP, and QP. In conversion, we used word level non-terminal nodes while benefiting from the headedness of Turkish language.

2.2 Algorithms

2.2.1 Rule-based Approach

While constructing an algorithm for constituency to dependency converters, there are some basic and simple steps that should be followed. The real hardship lies in creating specific rules for the needs of each language. Turkish has a rich morphology, with many suffixes and different cases carrying semantic or syntactic information. This made it compulsory to include in the algorithm some information from a morphological analyzer. As our input dataset already had each word morphologically analyzed through a semi-automatic process (Yıldız et al., 2019), we had the opportunity to extract information from their analyses.

The first problem to tackle is determining a head for each phrase in the phrase structure trees. Whereas in phrase structure trees the head of a constituent is not transparent due to its non-binary representation and linking style, in dependency structures the head and its dependents are always marked. This requires a head to be determined for each constituent. The most efficient way to achieve this is by constructing a hierarchical table which allocates a number to each type of phrase and listing them according to their prioritization. Table 1 contains the hierarchical head table used for the Turkish Penn Treebank. Hierarchical head table allocates priority numbers (from 1 to 5, with 1 having the top priority and 5 having the lowest) to different phrase labels. In a node with more than one daughter, the one with the lowest number becomes the head. In a case where phrases with the same priority number are sisters, the one on the right becomes the head.

The hierarchy of the tags are decided by considering the behaviors of the UD tags and the head final structure of Turkish. For instance words with VP/NOMP labels are always the root of the sentence. Therefore, they have the highest priority for being labeled as the head. The following group is the adjunct category. When these types of words are linked to the root, they are always dependents. The third category is the post positions and they can be linked to a word that is in the first or in the second category in the hierarchy, either case it is always a dependent. The fourth category consists of determiners and numeral modifiers. These are always dependents to noun phrases, therefore it is crucial that they are lower in the hierarchy. The last category of the hierarchy consists of the tags that are always dependents.

This table allows for an accurate head determination, especially in exceptional cases. Due to its head-final nature, in Turkish, heads of phrases generally appear on the right, thus at the end of the phrase. To write such an algorithm would be perfectly simple if it weren’t for some exceptions such as postpositions and negation. For instance, in the phrase *senin içın* ("for you") the postposition *için* is the rightmost word which should be the head according to a head-final algorithm. However, in UD annotations, the adpositions should be dependents of NPs, so the leftmost element *senin* (NP) should be head and the postposition
An NP is found as a sister to a PP, the NP has the priority to be the head. Another example to this can be the labels NOMP (nominal predicate) and NEG (negation), which appear as sisters with this respective order. However, the negation, even though it appears on the right of the main predicate, cannot be the head according to UD annotations. By allocating the number 5 to NEG and 1 to NOMP, we make sure that whenever NOMP appears with NEG, the algorithm will choose NOMP as the head and NEG as its dependent.

After the heads were determined, certain rules had to be written to link structures and phrase labels of the constituency trees with UD annotation tags. Figure 2 is a list which illustrates some of these conversion rules. There are 21 conversion rules in total, excluding the DEP rule which is used whenever the listed conversion rules fail to apply. The rules take the POS information of the head and dependent nodes in the constituency tree along with the morphological information of the word tokens. For instance, the sentence “hızlı koşuyor” meaning “s/he is running fast.” consists of an adverb and a verb. The tag hierarchy marks the verb “koşuyor” as the head and the adverb “hızlı” as the dependent. Then, the rules in Figure 2 apply and determine the relation between them. There are two labeling options for the relation between an adverb and a verb: ADVCL or ADVMOD. The difference between the two relations is that the ADVCL relation is used when the dependent is clausal, otherwise the adverb relation is labeled with ADVMOD. The clausal adverbials have a verbal root in their morphology. Therefore, the algorithm questions the existence of a verbal root in an adverb to determine the correct dependency relation. The adverb “hızlı” does not have a verbal root, so it is marked as ADVMOD. If it had a verbal root as in “hızlanarak”, meaning “in an increasing speed”, then, it would create an adverbal clause. Therefore, the algorithm would mark it as the ADVCL.

2.2.2 Machine Learning Approach

In training a machine learning algorithm, there are two main issues to consider: determining the head and how it is linked to its dependents, and determining the dependency relations between the head and its dependents. In order to achieve this, first, the algorithm sees a tree as a whole. Then, it finds the correct constituents by scanning the nodes of the tree. Once a node is matched, it cannot be overwritten by the upper nodes. This procedure continues until there are no more nodes left in the tree. Once the algorithm has established the constituents, we use an oracle which is the mechanism that determines the correct linkings between the heads and their dependents as well as the dependency relation types between them. We use two different oracles to find these constituents. One of them is the rule-based basic oracle and the other is a classifier oracle which is used as a machine learning model. After the constituents have been determined, they are put together to form a sentence.

The classifier oracle forms an instance list by taking the Parts of Speech (POS) tags of the words in the constituent and assigns them an index according to its position in the constituent. In another iteration, the head of the constituent is not included in the ranking. For instance, in a con-
if dependentNode == ADVP:
    if isVerbal(dependentWord):
        return ADVCL
    elif isNominal(dependentWord):
        return NMOD
    else:
        return ADVMOD
elif dependentNode == ADJP:
    if headNode == NP:
        if isVerbal(dependentWord):
            return ACL
        else:
            return AMOD
    return AMOD
elif dependentNode == PP:
    if headNode == NP:
        return CASE
    else:
        if isNominal(dependentWord):
            return NMOD
        else:
            return ADVMOD
elif dependentNode == NP:
    if isCompound(dependentWord)
        and isCompound(headWord):
            return COMPOUND
        if isProper(dependentWord):
            return FLAT
        return NMOD
elif headNode == VP:
    if isAccOrNom(dependentWord):
        return OBJ
    return OBL
elif dependentNode == S:
    if headNode == VP:
        return CCOMP
elif dependentNode == DP:
    return DET
elif dependentNode == NUM:
    return NUMMOD
elif dependentNode == INTJ:
    return DISCOURSE
elif dependentNode == NEG:
    return NEG
elif dependentNode == CONJP:
    return CC

Figure 2: Python rules for linking structures and phrase labels of the constituency trees with UD annotation tags

<table>
<thead>
<tr>
<th>Length</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.32</td>
</tr>
<tr>
<td>3</td>
<td>6.11</td>
</tr>
<tr>
<td>4</td>
<td>6.77</td>
</tr>
<tr>
<td>5</td>
<td>8.86</td>
</tr>
<tr>
<td>6</td>
<td>12.57</td>
</tr>
<tr>
<td>7</td>
<td>11.66</td>
</tr>
</tbody>
</table>

Table 2: Error rates according to constituency length (Random Forest, ensemble size = 50)

As for the second problem of choosing dependency relations, the first task was to assign the POS-tags and class information flags of heads and its dependents for the entirety of the instance list. For instance, in a constituent with an adjective and a noun, the instance would be ADJ NOUN 0 1. Here, the ADJ marks the POS tag of the first word of the constituent, the NOUN marks the second word, and the numbers mark their indexes. The instance lists are classified according to the length of the constituent. Later, the index of the head of the constituent is converted into class information. Thus, an instance like “ADJ NOUN 0 1” turns into ADJ NOUN 1 given that the head of this pair is the NOUN.

Subsequently, we have tested decision tree, Naive Bayes, KNN, and Random Forest models by using constituency and dependency treebanks of the same corpus. Running the models separately and as ensemble resulted in similar error rates. Thus, we decided to use the Random Forest which had the least errors. We tested the Random Forest model with several parameters and chose the one that minimizes the error rates. Table 2 shows error rates according to the constituency length. The error rates get higher as the constituency length increases. Therefore, we did not include constituents bigger than 7 words length and applied a rule-based algorithm for them.

As for the second problem of choosing dependency relations, the first task was to assign the POS-tags and class information flags of heads and its dependents for the entirety of the instance list. For instance, in a constituent with an adjective and a noun, the instance would be ADJ NOUN AMOD. Here, the first two tags mark the POS tag of the two words in linear order, and the last one marks the type of the dependency relation by considering the information provided by the POS tags.

However, the first results achieved by the random forest model were not satisfying. So as to achieve better performance, each word, including the head, was evaluated according to the following questions concerning their morphological
structure:
What is the POS-tag of the root?
Does the word have an ablative tag?
Does the word have a dative tag?
Does the word have a genitive tag?
Does the word have a nominative tag?
Does the word have an accusative tag?
Does the word have a proper tag?
Does the first word in the constituent share the same sense ID in Turkish WordNet with the head of the constituent?

These questions result in a sequence of true/false outputs along with the POS-tag information of the head of the constituent and its dependents, in addition to the class information provided at the end of the instance. The performance was satisfying after the implementation of the questions. The following string illustrates an instance of an ADJ-NOUN constituent which consists of two words. The string begins with the POS tag information of the first word of the constituent and it is followed by the true/false outputs relevant to this word. Then, it continues with the POS tag information of the second constituent and its applicable true/false outputs. The last "false" output stands for the last question and shows that the two words do not share the same word ID. The ADJP and NOMP marks the node names of the constituents in the phrase tree. Lastly, the AMOD label shows the relation between the two words.

ADJ false false false false false false
NOUN false false false true false false
false
ADJP NOMP
AMOD

3.2 Results & Discussion

Table 3 and 4 show the LAS, UAS and LA scores of our converted dependency treebank for the two approaches. The scores are calculated according to the amount of changes made by the human annotators on the converted dependency treebank. It should also be noted that the output sentences were all projective in both conversion methods.

Since this is the first converter for the Turkish language, we were unable to compare our results with other Turkish conversion tools. However, the results show that the rule-based approach lags behind the conversion studies held in other languages. The relatively lower LA score of the
Table 4: LAS, UAS, and LA parsing scores of the Machine Learning algorithm

<table>
<thead>
<tr>
<th>Metric</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAS</td>
<td>72.83</td>
</tr>
<tr>
<td>UAS</td>
<td>84.82</td>
</tr>
<tr>
<td>LA</td>
<td>78.18</td>
</tr>
</tbody>
</table>

A rule-based approach reflects the fact that confusion is existent between similar tags which cause debates even among the human annotators. On the other hand, the relatively higher UAS score of the rule-based algorithm shows that our head finding algorithm works better than the overall performance of the converter. This might be the outcome of the corpus we used since it consisted of formal sentences with mostly canonical order. A less formal corpus could give different results since the amount of scrambling would be different.

Nevertheless, the rule-based approach is not as successful as the machine learning algorithm. This is an interesting finding given that the machine learning approach resulted in better scores even though the rule-based algorithm was specifically tailored for the requirements of the Turkish Language. The results indicate that dividing the conversion problem into two phases as head finding and determining the dependency relation is a better methodology for conversion studies because it not only results in better scores but also provides a more adaptable algorithm that can be used for other languages. This supports the idea that there is confusion between similar tags, and this affects the performance of the converter. The main reasons for the confusions have been explained below.

Tables 5 and 6 shows the confusion matrices of the 13 most frequent tags used in the data for rule-based and machine learning algorithms respectively. From this table, some of our mistakes can be explained as a result of the co-occurrence of the same lexical items with different functions in Turkish. For instance, the word yumuşak meaning "soft" can occur as an adjective as in yumuşak yastık meaning "soft pillow", as an adverb modifying a verb in a sentence like "Yumuşak davranın," meaning "S/he acted softly." It can also function as an adjectival predicate to be labelled as a root as in “Onun kumaşı yumuşak,” meaning “Its cloth is soft”. Notice that the three words with different functions do not show any morphological variance on the word. These kinds of structures affect the labeling of ADVMOD, AMOD, OBL, and NMOD the most. Table 6 shows this effect. For example, 5.6% of OBL had been mistakenly labelled as ADVMOD and another 34.8% of them were labelled as NMOD. Considering the syntactic position helps to differentiate between these tags to some extent, especially in OBL-ADVMOD cases. However, understanding the function of morphologically identical structures is still an important issue because of the flexible word order. Assumptions on syntactic position often cause problems as a result of the flexible word order of Turkish language. An example is the NSUBJ tag. The table shows that 25.3% of the total mistaken tags for NSUBJ consists of the NMOD tags and another 16.4% consists of the OBJ tags. Some of these mistakes can be explained by scrambling. In the canonical word order, we would expect the first NP of a sentence in Turkish to be the subject as in the treebank all the complements of the verb were put under the predicate node. However, there were sentences where other NP structures occurred before the subject. As a result, both flexible word order and the morphological reasons mentioned above gave rise to the confusion among these tags in the rule-based algorithm.

Table 6 shows that the machine learning algorithm prioritizes the learning of the more frequent tags such as AMOD, ADVMOD, NMOD and so on in that it shows higher performance than the rule-based algorithm over these tags. This proves that the machine learning approach does not tackle the same confusion problems as the rule-based model stated above. On the other hand, there are areas in which the machine learning algorithm performed poorer than the rule-based algorithm such as identifying FLAT and CASE relations. These tags share the common property of reversed attachment type. This uniqueness of these relations can easily be applied in rule-based algorithm but apparently it causes confusion to the machine learning models.

All of these examples show that machine learning approach is helpful to increase the parsing accuracy for languages with rich morphology and flexible word order since it takes each sample as an input, and evaluates the patterns according to the relevant information provided by the data rather than trying to fit the data into a structure. Both the rule-based model and the machine learning model
employ the use of dependency relations which are standardized according to the UD annotation scheme. In a different domain, the frequency of the labels used can change slightly. However, as the edge cases are a result of the structure of the language, a domain change would not significantly affect the accuracy difference between the two models. Therefore, the scores presented in this study are not domain specific. In addition, it should be noted that the comparison of the models are based solely on the accuracy. The two models require different kinds of prior work. For instance, machine learning approach requires an annotated corpus to be trained on while the rule-based model does not need this kind of a training data. This is one of the reasons that all of the previous conversion studies employed a rule-based approach as there was not any data to train the converter in most languages. However, there are now enough training corpora to employ machine learning approach for further studies.

4 Conclusion

Overall, this project presents a remarkable contribution to UD studies in Turkish, in that it introduces the first constituency to the dependency conversion tool. Furthermore, the performance comparison which was provided shows that machine learning algorithms are able to achieve better accuracy scores compared to rule-based approaches. We hope that this tool will prove to be useful in future studies on Turkish dependency corpora as well. Thanks to the rapid transformation from phrase structure trees, it will allow for the creation of more dependency data, paving the way for better Turkish dependency parsers.

Moreover, as a result of this conversion, a corpus of 10,000 new sentences has been added to the Turkish dependency corpora, constituting one of the largest dependency corpora in Turkish. Besides, considering that the Penn Treebank constituency trees have previously been transformed into dependency trees with the help of many conversion tools such as the Stanford conversion tool (Marneffe et al., 2006) and ClearNLP (Choi and Palmer, 2012) the output we provide represents an impeccable overview of comparative structures of English and Turkish UD trees, thus offering a cross-linguistic perspective.
References


J. D. Choi and M. Palmer. 2012. Guidelines for the clear style constituent to dependency conversion.


Abstract

In the social media, users frequently use small images called emojis in their posts. Although using emojis in texts plays a key role in recent communication systems, less attention has been paid on their positions in the given texts, despite that users carefully choose and put an emoji that matches their post. Exploring positions of emojis in texts will enhance understanding of the relationship between emojis and texts. We extend an emoji label prediction task taking into account the information of emoji positions, by jointly learning the emoji position in a tweet to predict the emoji label. The results demonstrate that the position of emojis in texts is a good clue to boost the performance of emoji label prediction. Human evaluation validates that there exists a suitable emoji position in a tweet, and our proposed task is able to make tweets more fancy and natural. In addition, considering emoji position can further improve the performance for the irony detection task compared to the emoji label prediction. We also report the experimental results for the modified dataset, due to the problem of the original dataset for the first shared task to predict an emoji label in SemEval2018.

1 Introduction

The advent of emojis has dramatically changed the style of human communication. Currently, emojis are widely used on social media platforms such as Twitter, Facebook, and Instagram. Owing to the prevalent use of emojis in the last few years, they become a target of recent researches.

As reported by Kralj Novak et al. (2015), exploring the interplay between emojis and texts can be a clue for better natural language understanding (NLU) in social media datasets with full of emojis. In response to this report, Barbieri et al. (2017) proposed a task to predict an emoji label given a text, which can contribute to various natural language processing (NLP) tasks such as information retrieval, social media content generation, and sentiment/emotion analysis. To solve this task, they used an LSTM-based tagger and achieved even higher performance than human prediction.

However, in spite of the success of the previous researches, the existing emoji prediction task targets only at predicting a single emoji given a text without considering its position. Kralj Novak et al. (2015) reported that social media users place an emoji differently based on its emoji type in a tweet. They analyzed 751 emoji labels by dividing them into positive, negative and neutral types to show that the type of an emoji strongly correlates with its position in a text. This indicates the information of the emoji position can be a good clue to boost emoji label prediction.

Hence, it is necessary to consider an emoji label and its position together. Social media users do not simply put an emoji at the end of a sentence. Automated systems of predicting an emoji with its position that consider surrounding linguistic information can further enhance the aforementioned...
tasks in the field of NLU and NLP. Figure 1 describes the difference between the previous task of emoji prediction and our proposed task of emoji insertion to texts. In the first and second examples, we can understand that the position of emojis is important and indispensable in cases where they are used instead of a word or a character. In the third example, the position of an emoji is required to investigate the relationship between the emoji and multiple sentences. To take into account the position of emoji labels, we first demonstrate that the information of the emoji position can improve the performance of emoji label prediction by using the gold emoji position.

We then extend the task of emoji prediction by proposing a novel task of inserting a suitable emoji into a suitable position in a given tweet. To solve this new method, we propose Bi-Affine-based and Bi-LSTM-based models that can jointly predict both emoji label and its position without relying on any linguistic features.

Class-imbalanced data is one of the major concerns in the field of emoji prediction (Barbieri et al., 2017) because emojis of positive emotion are more frequently used on social media and dominate minor classes of negative and non-emotional emojis (Kralj Novak et al., 2015). It is difficult to obtain better results for imbalanced emoji classes. Figure 2 shows the imbalanced frequencies of emojis that obey a power-law distribution. To deal with the imbalanced frequencies of emojis, we also propose a novel method, Contextualized Dynamic-Smoothing (CDS), which can be adopted in our proposed models.

Due to the problems of the original dataset for the shared task for multilingual emoji prediction in SemEval2018 task2 (Barbieri et al., 2018b), that sometimes mistakenly contains “traces” of emojis in tweets, we prepared the modified dataset for our experiments.

Experimental results show that incorporating the gold emoji position can improve the performance of emoji label prediction. In addition, our proposed models of jointly predicting an emoji label and its position outperform the baseline model for emoji label prediction that does not consider the emoji position in terms of F-1 scores. Specifically, the results of our models with CDS, that solves the problem of the imbalanced emoji frequency, show the additional improvement on infrequent and non-emotional emojis. Human evaluation shows that there is a suitable emoji position in a tweet. In addition, our proposed task to insert an emoji in a tweet can actually make the tweet more fancy and natural, compared to the current emoji prediction task. Considering emoji positions can further improve the performance on the irony detection task compared to the current emoji prediction.

2 Related Work

Social media platforms contain various types of emotion expression methods such as emojis and kaomojis (Kwon et al., 2019). Recently, emojis occupy a large proportion on social media owing to the richness in their information. Emoji information can enhance the quality of social media datasets for researches such as sentiment, irony, emotion, and sarcasm analysis (Felbo et al., 2017; Singh et al., 2019). Although emojis can be interpreted differently based on social media platforms (Miller et al., 2016), their usage is similar in multiple countries (Barbieri et al., 2016), meaning that they can be predicted in different languages such as English, Spanish (Barbieri et al., 2018b), and Italian (Ronzano et al., 2018). Emoji prediction can be boosted by incorporating not only texts but photos in the Instagram dataset (Barbieri et al., 2018a) and can directly improve the task of sentiment analysis (Chen et al., 2019). Also, emoji prediction can be used in dialogue systems to recommend a suitable emoji (Xie et al., 2016), and using emojis in chatbot systems is effective to attract users, specifically in the conversation for mental wellbeing (Fadhil et al., 2018). Recently, Ma et al. (2020) released an emoji label prediction dataset with passage-level multi-class/multi-label, and aspect-level multi-class annotations.
Our work to predict both an emoji label and its position is in accordance with sequential labeling tasks. In dependency parsing, the Bi-Affine dependency label classifier was proposed to study the relationship between two words, and achieved the state-of-the-art performance (Dozat and Manning, 2017). Currently, Bi-Affine layers have been widely adopted in various tasks, such as relation extraction (Nguyen and Verspoor, 2019), mention detection and clustering (Zhang et al., 2018), owing to their effectiveness. Because they have demonstrated the standout performance in the aforementioned tasks, we employ them to learn emoji positions to predict emoji labels.

3 Models

We introduce our proposed models for jointly predicting an emoji and its position in each tweet. Bi-LSTM networks are first used to encode a given tweet (Graves et al., 2013) and a Bi-Affine layer is used to predict the position of an emoji. Then, the predicted emoji position is used to predict the label of the emoji with CDS.

3.1 Bi-LSTM

Bi-LSTM networks are used to capture the forward and backward context of the input text, given words of input text \( S = \{w_0, w_1, w_2, ..., w_N, w_{N+1}\} \), which are encoded as word embeddings \( \{e_0, e_1, e_2, ..., e_N, e_{N+1}\} \). \( w_0 \) and \( w_{N+1} \) indicate “<s>” and “</s>” tokens for the beginning and end of a sentence, respectively. Then, these word embeddings are fed to the forward and backward LSTMs and are converted to forward and backward hidden states \( h_f \) and \( h_b \), respectively. We use the concatenated hidden state \( h_t = [h_f; h_b] \) as the output of Bi-LSTM for each time step.

3.2 Selective Gate

To enhance the information of each hidden state for the eligible emoji label and its position, we incorporate an selective module (Zhou et al., 2017) on top of the Bi-LSTM networks. Specifically, each \( h_t \), which is the output of Bi-LSTM, is passed through the selective gate, \( sGate_t \). The \( sGate_t \) decides the importance for each hidden state by considering the entire sentence information that consists of \( s = [h_0; h_{N+1}] \), where \( h_0 \) is the first backward hidden state and \( h_{N+1} \) is the last forward hidden state. The formula for \( sGate_t \) is defined as follows:

\[
\begin{align*}
    sGate_t &= \sigma(W_h h_t + W_s s + b), \\
    h_t' &= h_t \odot sGate_t,
\end{align*}
\]

where \( W_h \) and \( W_s \) indicate weight matrices, \( b \) is the bias term. \( \odot \) is a symbol for element-wise multiplication and \( \sigma \) is a sigmoid activation function for normalizing ranges of gate outputs. The sequence of hidden states, \( \{h_0, h_1, h_2, ..., h_N, h_{N+1}\} \), is computed through the selective gate. Then, a new sequence of embeddings, \( \{h_0', h_1', h_2', ..., h_N', h_{N+1}'\} \), which takes into account the sentence information for each time step, is generated.

3.3 Position Module

This module predicts an emoji position, \( p\text{os} \). The emoji position is predicted by the following equation:

\[
\text{p\text{os}} = \arg \max_{\text{pos}} \text{score}_p(\text{pos}),
\]

where \( \text{pos} \) is possible emoji positions. To calculate the position score, \( \text{score}_p(\text{pos}) \), we propose the following two networks.

3.3.1 Simple Concatenation

We simply use the concatenation of the neighboring two hidden states, \( [h_i', h_{i+1}'] \), to calculate each position score as follows:

\[
\text{score}_p(\text{pos}) = v_p t_i,
\]

\[
t_i = \sigma(W[h_i'; h_{i+1}'] + b),
\]

where \( W \) is the parameter matrix, and \( b \) is the bias term. \( v_p \) is the learnable vector for emoji position prediction. \( \sigma \) is the activation function of the Rectified Linear Unit (Nair and Hinton, 2010). The function, \( \text{score}_p \), returns a scalar value for each possible emoji position.

3.3.2 Bi-Affine

The simple concatenation approach can explain the interaction between the two hidden states, \( h_i' \) and \( h_{i+1}' \). However, the clues from a single state without any interaction with the other state might also be informative for emoji position prediction. To facilitate extraction of such pure clues, we additionally propose a method to use Bi-Affine-based transformation of the hidden states \( h_i' \) and \( h_{i+1}' \), which contains the terms only with \( h_i' \) or \( h_{i+1}' \). Specifically, instead of using Equations (4) and (5), the
Bi-Affine function is used to calculate each position score as follows:

\[
\text{score}_p(\text{pos}) = \text{biaffine}(h'_i, h'_{i+1}), \quad (6)
\]

\[
\text{biaffine}(h_i, h_r) = v^T_i h_i + h^T_r W_3 h_r + v^T_r h_r, \quad (7)
\]

where \(v^T_i\) and \(v^T_r\) are learnable vectors. \(W_3\) is the weight matrix.

3.4 Label Module

As a downstream procedure, this module predicts an emoji label, \(\text{label}\). The emoji label is predicted by the following equation:

\[
\text{label} = \arg \max_{\text{label}} \text{score}_l(\text{label}), \quad (8)
\]

where \(\text{label}\) is emoji class labels. Following two types of networks were used for modeling the score function, \(\text{score}_l(\text{label})\), to predict an emoji label.

3.4.1 Linear Projection

We used a linear projection layer to map the simple concatenated hidden states in Equation (5) to the emoji label space. To take the predicted emoji position into account in predicting an emoji label, \(\hat{p}\) is applied to calculate the score function:

\[
\text{score}_l(\text{label}) = W_l t_{\hat{p}} + b, \quad (9)
\]

where \(W_l\) and \(b\) are the weight matrix and the bias term for emoji label prediction, respectively.

3.4.2 Contextualized Dynamic-Smoothing (CDS)

As shown in Figure 2, the frequencies of emojis are imbalanced, that can result in difficulties to obtain better performance in emoji label prediction (Barbieri et al., 2017). To address this issue, we propose a novel smoothing method based on the left and right context vectors for the emoji label prediction. In Equation (9), the bias term \(b\) for predicting an emoji label is shared in all emoji positions regardless of the predicted emoji position \(\hat{p}\). It would be beneficial for the bias term to consider the eligible value at each emoji position to predict an emoji label. Hence, we calculate the contextualized bias term based on the information of each emoji position using Bi-Affine layers, revealed as follows:

\[
b_{\text{dynamic}} = h'^T_l W_4 h'_r, \quad (10)
\]

where \(W_4\) is the weight matrix. We use \(h'_l\) and \(h'_r\) to obtain various bias values as the effect of smoothing the distribution of emoji classes. \(b_{\text{dynamic}}\) is added to calculate the emoji label score as follows:

\[
\text{score}_l(\text{label}) = W_l t_{\hat{p}} + b_{\text{dynamic}}. \quad (11)
\]

3.5 Objective Function

Our objective function is composed of two Cross-Entropy losses for the emoji label and its position. To learn parameters by backpropagating two losses, we sum two losses through a hyperparameter \(\lambda\):

\[
\text{loss} = \text{loss}_{\text{label}} + \lambda \cdot \text{loss}_{\text{pos}}. \quad (12)
\]

4 Experimental Settings

In this section, we report the problems of the original datasets from the SemEval2018 task2. Then, we describe the datasets we modify and use in predicting emoji labels and their positions.

4.1 Datasets

4.1.1 Original SemEval2018 datasets

The SemEval2018 task2 was the first shared task in predicting emoji labels with multilingual datasets: English and Spanish tweets (Barbieri et al., 2018b). Although various submitted systems attained good results in predicting emoji labels, we found the released SemEval2018 task2 datasets have some problems.

The trial and test datasets were composed of 50k tweets and were released by dividing them into two files, target emoji labels and their corresponding input tweets. However, some of English and Spanish input tweets contained hidden “traces” of the corresponding target emoji labels. Specifically, English input tweets in the test dataset whose target emoji labels are \(\heartsuit\) and \(\star\) had such hidden “traces” over than 85% and 98%, respectively. They essentially consisted of two Unicode fragments of emojis: “\ufe0f” and “\ufe0e”. They indicated the target emojis with their exact positions in the input tweets. Moreover, the released code, which splits the raw training dataset into the two files of input tweets and target emoji labels in the preprocessing step, also had a problem. It affects the training dataset to contain such hidden “traces”.

The emoji \(\heartsuit\), including the “traces”, is the most frequent in all of the training, trial and test datasets. It may influence the performance of emoji label prediction.

4.1.2 Modified Dataset

To remedy the above problems, we modified the shared English dataset from the SemEval2018
task2. We refrained from using the original trial and test datasets owing to no information of emoji positions and the problems of hidden “traces” in them. In addition, we did not use the released code, which divides the raw training dataset into two files of the input tweets and target emoji labels.

Thus, we retrieved 450,021 tweets in the training dataset. To use exact positions of emojis, we discarded a few tweets with an emoji appearing more than once (e.g., My baby bear 😍) from the extracted training dataset. As a result, we obtained our final dataset of 357,305 tweets in total with 20 emoji class labels. Figure 2 indicates the statistics of emojis in the final dataset. We also prepared its subsets for the 10 (250,747 tweets) and 5 most frequent emojis (180,767 tweets), with the same procedure in the previous work for emoji label prediction (Barbieri et al., 2017, 2018a). We randomly shuffled and split them into three parts: training (80%), validation (10%), and test (10%) datasets.

4.2 Baseline Model

Barbieri et al. (2017) proposed the task of emoji label prediction with a simple emoji classifier. Their Bi-LSTM classifier received an input tweet without an emoji label to predict a suitable emoji label. In this study, we used their network as our baseline model to examine whether the information of emoji positions can improve the performance of emoji label prediction.

4.3 Parameter Settings

We used one-layer stacked Bi-LSTMs with hidden states of 256 dimensions for all models. The Adam optimizer was used with a learning rate of 0.001, betas of 0.9 and 0.999, and epsilon of $1.0 \times 10^{-8}$. The value of $\lambda$ in Equation (12) is 0.2, which was estimated based on the performance of the average macro F-1 score in emoji label prediction on the validation dataset. Glove embeddings are well-known pre-trained vector representations for words (Pennington et al., 2014). We used 27 billion pre-trained word embeddings of 200 dimensions trained from the Twitter dataset. We concatenated Glove embeddings with the initialized word embeddings to obtain each word embedding.

5 Experiments and Evaluation

5.1 Emoji Insertion

5.1.1 Emoji Label Prediction

As the evaluation metrics, we employed the precision, recall, and macro F-1 scores, officially used in the SemEval2018 task2.

In Table 1, we compare the results of emoji label prediction for the baseline and proposed models with the gold and jointly learned emoji positions, respectively. The baseline was described in Section 4.2. Upper Bounds 1 and 2 used the gold emoji positions to predict emoji labels without and with applying CDS, respectively. The results for Upper Bounds 1 and 2 demonstrate that the information of emoji positions can improve the performance of emoji label prediction. Specifically, using gold emoji positions in the model of Upper Bound 1 improved +3, +4.41, and +2.36 average F-1 scores on the modified datasets for top 5, 10, and 20 emojis, compared with the Baseline that does not consider emoji positions. Incorporating the proposed CDS to deal with the class-imbalanced problem yields the further improvement of +1.22, +1.14, and +1.57 on each dataset between Upper Bounds 1 and 2.

Joint Learning models 1 and 2 predicted the position of emojis using simple concatenation and Bi-affine layers, respectively. Joint Learning models 3 and 4 incorporate CDS to predict emoji labels. The joint learning models without any external resources outperformed the Baseline model. The highest F-1 score was obtained by Joint Learning 4, that uses Bi-Affine layers to predict the position of emojis and applies CDS to predict the emoji labels. It improved +2.77, +2.5, and +1.93 average F-1 scores on the modified datasets for top 5, 10, and 20 emojis, compared with the Baseline.

Table 2 describes F-1 scores for top 20 individual emojis with the Baseline and Upper Bound 2. Applying CDS with the position information can improve the performance of predicting non-emotional emojis. The performance for the non-emotional emojis, 😐, 😄, 😎, 😘, 😍, 😂, 😊, increased by +7.7, +2.6, +3.5, +2.5, +5.3, +4, and +5.3, respectively, compared with the baseline model. In addition, incorporating CDS with the information of emoji positions is effective for infrequent emojis, 😞 (+2.5), 😓 (+5.3), 😞 (+10.2). It also helps the model to predict 😞 and 😞 with F-1 scores of 5 and 3, respectively.
Table 1: Average F-1 scores for emoji label prediction and accuracy for emoji position prediction. The best scores in all the models except Upper Bounds are shown in bold. † indicates that the improvement from the Baseline is statistically significant by using the paired bootstrap resampling method (Koehn, 2004) (p < 0.001).

Table 2: F1 scores for emoji label prediction with top 20 emojis. Base. indicates the baseline model and UB2. describes the Upper Bound 2. % is the proportion of each emoji in the test dataset.

Table 3: Examples of predicting emoji labels and their positions with our best model, Joint Learning 4, compared with the baseline model.
rank-probability from Joint Learning 3 shows that the probability distribution of emoji positions is in accordance with emoji labels, meaning that the probability distribution is not uniform. Thus, the emoji position is predictable because the emoji label is predictable (Barbieri et al., 2017). Moreover, the probability distributions of emoji labels from Joint Learning models 3 and 4 demonstrate that our proposed CDS successfully smoothed estimated probabilities of emoji labels, and the probabilities of low ranked emoji labels increased.

**Algorithm 1** Construction of the human evaluation dataset.

```
Require:
Tweets T = \{T_1, T_2, ..., T_n\}
tweetNLP = part-of-speech tagger
Initialize the filtered tweets, F = [ ]
for i = 1 to n do
  tagged ← tweetNLP(T_i)
  v ← tagged.count(‘v’), o ← tagged.count(‘o’)
  if v > 2 and o ≥ 2 then
    F.append(T_i)
  end if
end for
return F
```

5.2 Human Evaluation

We carried out an experiment to show that there is a suitable emoji position for individual emojis in tweets. Then, we conducted an experiment in which we compared current and our proposed tasks.

To encourage and guide the human evaluation, we applied a filtering step to harvest suitable tweets from the SemEval2018 training dataset. Algorithm 1 describes how to filter tweets. We first applied a tweetNLP, which is a part-of-speech tagger for tweets that can take into account emoji information (Gimpel et al., 2011). It can annotate part-of-speech tags to emojis. Specifically, emojis can be tagged with “v” (Verb) when they are used instead of a word (e.g., ‘ is frequently used instead of a word “love”). Thus, we inserted rules that the number of “o” (Pronoun) and “v” (Verb) tags is greater than two to extract tweets including an emoji used instead of a word and tweets with multiple sentences.

To show there exist a suitable emoji position, we constructed a dataset, **Gold (Inside)**, by randomly selecting 1,000 tweets that emojis are not located at the end from the filtered dataset. We prepared a **End** dataset by moving emojis to the end of the tweet from the **Gold (Inside)** dataset. Because the gold emoji position can be end of the tweet, we also constructed a **Gold (End)** dataset by randomly selecting 500 tweets that emojis are located at the end from the filtered dataset. We also prepared a **Random (Inside)** dataset by randomly relocating emojis in each tweet from the **Gold (End)** dataset. Then, we constructed a **Current (End)** and **Proposal** datasets by applying Baseline model (See. 4.2) and our best model, Joint Learning 4, on the modified test dataset. Because Baseline model cannot insert an emoji in a tweet, we attached a predicted emoji at the end of a tweet. **Current (End)** and **Proposal** datasets consist of randomly selected 500 tweets after the filtering step.

We asked humans to select better tweets to demonstrate that there exists a suitable emoji position and our proposed task can contribute to making tweets more fancy and natural compared to the current emoji prediction task. After presenting two tweets from **Gold (Inside)** and **End** datasets, **Gold (End)** and **Random (Inside)** datasets, and **Current (End)** and **Proposal** datasets, we asked two questions for the human annotators “Which tweet is more fancy?” and “which tweet is more natural?”’. The crowd-sourcing platform, “Mechanical Turk”, was used to design an experiment. The same tweets were displayed to five human annotators\(^1\) and the final decision was selected based on majority agreement.

Table 4 shows results of human evaluations between the gold emoji position and relocated emoji position. When the gold emoji position is inside of tweets, it can make fancy and natural tweets more than 90% and 87%, respectively, compared to moving emojis to the end of tweets. In addition, when the gold emoji position is end of tweets, it can make fancy and natural tweets more than 73% and 78%, respectively, compared to randomly relocated.
cating emoji in tweets. Thus, individual emojis are not always located at the end of tweets and there exists a suitable emoji position in tweets. To assess the reliability of the agreement between the human annotators and their decision, we used a statistical measure called Fleiss’ Kappa (L. Fleiss, 1971). We obtained Kappa scores of 0.52 for Fancy and 0.68 for Natural between Gold (Inside) and End datasets, which indicate moderate and substantial agreement, respectively. In addition, we obtained Kappa scores of 0.41 for Fancy and 0.26 for Natural between Random (Inside) and Gold (End) datasets, which indicate moderate and fair agreement, respectively. Thus, considering emoji position is necessary to understand social media texts. As can be seen in Table 5, our proposed task can actually make tweets be more fancy and natural compared to the current task.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Fancy</th>
<th>Natural</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold (Inside)</td>
<td>92.0</td>
<td>87.5</td>
</tr>
<tr>
<td>End</td>
<td>80</td>
<td>127</td>
</tr>
<tr>
<td>Total</td>
<td>1000</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 4: Human evaluation between gold emoji position and relocated emoji position.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Fancy</th>
<th>Natural</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random (Inside)</td>
<td>135</td>
<td>110</td>
</tr>
<tr>
<td>Gold (End)</td>
<td>365</td>
<td>390</td>
</tr>
<tr>
<td>Total</td>
<td>500</td>
<td>500</td>
</tr>
</tbody>
</table>

Table 5: Human evaluation between current task and proposed task.

5.3 Irony Detection

To validate the effectiveness of considering emoji position to understand social media texts, we conducted the downstream task, irony detection. We used the 2-class (ironic or not ironic) and 3-class (ironic by clash, other, or not ironic) raw irony detection datasets released by the SemEval2018 Task 3A and 3B, respectively (Van Hee et al., 2018). The macro-average F1 score was used to evaluate the performance by following previous work. Because raw datasets were provided with training and test sets only, we randomly shuffled and split the training dataset into two parts: training (80%) and validation (20%). For each raw dataset, we constructed Emoji w/o position and Emoji w/ position datasets by inserting an emoji using Baseline and Joint Learning 4 models. Because Baseline cannot insert an emoji, we simply attached a predicted emoji at the end of the sentence. Thus, three datasets of the released raw, constructed Emoji w/o position, and Emoji w/ position were used to perform irony detection.

Table 6 shows the results. The first block shows the reported scores of the top 2 previously submitted systems on the SemEval2018 Task 3A and 3B. We used the second best model, NTUA-SLP, which used Bi-Lstm with the self-attention networks. The second block shows the re-run performances using the NTUA-SLP model\(^2\) with raw, Emoji w/o position, and Emoji w/ position datasets. Using the Emoji w/o position dataset improved the performance of irony detection compared to using the raw dataset in both task3A and 3B. In addition, using Emoji w/ position dataset on the NTUA-SLP model outperformed using Emoji w/o position dataset and achieved better performances compared to the best models of the THU-NGN and UCDCC in both task3A and 3B. This improvement can be explained by the fact that social media users put an emoji carefully to convey their meaning and the position of emoji is effective to understand social media texts.

<table>
<thead>
<tr>
<th>Team</th>
<th>F-1</th>
<th>Team</th>
<th>F-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>THU-NGN (Wu et al., 2018)</td>
<td>70.5</td>
<td>UCDCC (Ghosh and Veale, 2018)</td>
<td>50.1</td>
</tr>
<tr>
<td>NTUA-SLP (Baziotis et al., 2018)</td>
<td>67.2</td>
<td>NTUA-SLP (Baziotis et al., 2018)</td>
<td>49.6</td>
</tr>
<tr>
<td>NTUA-SLP + raw</td>
<td>69.7</td>
<td>NTUA-SLP + raw</td>
<td>48.9</td>
</tr>
<tr>
<td>NTUA-SLP + Emoji w/o position</td>
<td>70.1</td>
<td>NTUA-SLP + Emoji w/o position</td>
<td>49.6</td>
</tr>
<tr>
<td>NTUA-SLP + Emoji w/ position</td>
<td>70.8</td>
<td>NTUA-SLP + Emoji w/ position</td>
<td>50.9</td>
</tr>
</tbody>
</table>

Table 6: The results of irony detection. The left and right table shows the performance on the SemEval2018 task3A (2-class) and 3B (3-class), respectively.

6 Conclusion

In this paper, we presented a novel task of jointly predicting the emoji label and its position by using Contextualized Dynamic-Smoothing (CDS) with Bi-Affine layers. The experimental results showed that the information of emoji positions is important and can improve the performance of emoji label prediction without external resources. Because the original public datasets of emoji label prediction had problems, we prepared the modified datasets. Human evaluation validated there exists a suitable emoji position in a tweet and our proposed task can make tweets more fancy and natural compared to the current emoji prediction. Moreover, considering emoji position is effective on the task of irony detection.

\(^2\)https://github.com/chaziotis/ntua-slp-semeval2018/
References


Hannah Miller, Jacob Thebault-Spieker, Shuo Chang, Isaac Johnson, Loren Terveen, and Brent Hecht. 2016. “blissfully happy” or “ready to fight”: Varying interpretations of emoji. In Proceedings of the


Addressing Slot-Value Changes in Task-oriented Dialogue Systems
Through Dialogue Domain Adaptation

Tiziano Labruna
Fondazione Bruno Kessler,
Trento, Italy
Free University of Bozen-Bolzano
tlabruna@fbk.eu

Bernardo Magnini
Fondazione Bruno Kessler,
Trento, Italy
magnini@fbk.eu

Abstract
Recent task-oriented dialogue systems learn a model from annotated dialogues, and such dialogues are in turn collected and annotated so that they are consistent with certain domain knowledge. However, in real scenarios, domain knowledge is subject to frequent changes, and initial training dialogues may soon become obsolete, resulting in a significant decrease of the model performance. In this paper, we investigate the relationship between training dialogues and domain knowledge, and propose dialogue domain adaptation, a methodology aiming at adapting initial training dialogues to changes intervened in the domain knowledge. We focus on slot-value changes (e.g., when new slot-values are available to describe domain entities) and define an experimental setting for dialogue domain adaptation. First, we show that current state-of-the-art models for dialogue state tracking are still poorly robust to slot-value changes of the domain knowledge. Then, we compare different domain adaptation strategies, showing that simple techniques are effective to reduce the gap between training dialogues and domain knowledge.

1 Introduction
Conversational agents are receiving great attention in recent years, both in research and applications (McTear, 2020), mainly because of the progress achieved by neural approaches in modeling dialogue phenomena (Louvan and Magnini, 2020), (Balaraman et al., 2021). Particularly, we focus on task-oriented dialogue systems (Young et al., 2010), which are able to assist a user for specific tasks (e.g., booking a restaurant, taking an appointment, execute commands) in a certain conversational domain. Such data-driven dialogue systems typically learn a model from annotated training dialogues (e.g., (Budzianowski et al., 2018), (Du et al., 2020), (Price, 1990)), and such dialogues, in turn, are collected and annotated according to a certain domain scenario (e.g., restaurants in a certain town, songs in a certain music data-set, etc.). Once the model is trained, it is applied to understand new conversations in the same domain, or in a similar domain.

However, in real scenarios, domain knowledge is subject to frequent changes, and initial training dialogues may soon become obsolete, resulting in a significant decrease in the model performance. It is common that new domains (e.g., RESTAURANT) are added or removed for a certain conversational scenario, as well as slots, slot-values and instances. Such situations require the capacity of the dialogue system to adapt its behaviours to domain knowledge changes. Depending on the complexity of the changes that occur, domain adaptation of data-driven systems can be approached in two directions: (i) improving the model robustness, and (ii) adapting the training dialogues to the new situation. While the first direction has been largely explored through several techniques, including transfer learning (Louvan and Magnini, 2019), and zero-shot learning though schema-guided models (Wu et al., 2019a; Kim et al., 2020; Zhang et al., 2019; Heck et al., 2020; Balaraman and Magnini, 2021), and delexicalization (Henderson et al., 2014a,b; Yu et al., 2020), in this paper we take the second, less investigated, perspective, focusing on the relation between training dialogues and domain knowledge.

We have defined a new experimental setting for dialogue domain adaptation, where, given an initial conversational domain (KB-SOURCE), available training dialogues (D-SOURCE) are adapted to be as much as possible consistent to a modified knowledge base (KB-TARGET), resulting...
in new set of dialogues (D-TARGET). Then, we use a state-of-the-art model for dialogue state tracking and assess the performance of different adaptation strategies against a gold standard of manually adapted target dialogues. We run a number of experiments showing that: (i) current state-of-the-art models are very poorly robust to changes over the domain knowledge; (ii) a particular class of domain changes, i.e., slot-value changes, can be effectively addressed through simple dialogue domain adaptation techniques, which operate substitutions over D-SOURCE. Finally, as part of our study, we highlight that current component-based evaluation settings for task-oriented dialogue systems (i.e., slot filling, intent detection, dialogue state tracking, utterance generation) are not sensible to correctness of system responses, which, instead, is crucial to assess domain adaptability.

The paper is structured as follows. Section 2 provides basic background in task-oriented dialogue systems. Section 3 defines the conversational adaptation task, while in Section 4 we introduce the proposed dialogue domain adaptation approach. Section 5 introduces the new experimental setting, and Section 6 provides the results of our experiments and discusses them.

2 Task-oriented Dialogue Framework

This section provides background related to task-oriented dialogue systems, particularly about how domain knowledge is managed and about dialogue state tracking.

2.1 Domain Knowledge

According to most of the recent literature (Budzianowski et al., 2018; Bordes et al., 2017; Mrkšić et al., 2017), we assume a task-oriented dialogue between a system and an user, composed of a sequence of turns \( \{t_1, t_2, ..., t_n\} \). The goal of the dialogue is to retrieve a set of entities (possible empty) in a domain knowledge base \((KB)\) that satisfy the user needs. A domain ontology \( O \) provides a schema for the \( KB \), and typically represents entities (e.g., RESTAURANT, HOTEL, MOVIE) according to a pre-defined set of slots \( S \) (e.g., FOOD, AREA, PRICE, for the RESTAURANT domain), and values that a certain slot can assume (e.g., EXPENSIVE, MODERATE and CHEAP, for the slot PRICE). On the basis of the entities defined in the domain ontology, the application knowledge base, \( KB \), is then populated with instances of such entities. As in most of the literature, we distinguish informable slots, which the user can use to constraint the search (e.g., AREA), and requestable slots (e.g., PHONE NUMBER), whose values are typically asked only when a certain entity has been retrieved through the dialogue.

At each turn in the dialogue, both the user and the system may refer to facts in the \( KB \), the user with the goal of retrieving entities matching his/her needs, and the system to propose entities that can help the user to achieve the dialogue goals.

2.2 Dialogue State Tracking

In a task-oriented system a dialogue state tracker (DST) maintains a distribution over the dialogue states based on the dialogue history. A dialogue state \( d_i \) for a turn \( t_i \) is typically represented as a set of slot-value pairs, such as \{PRICE=Moderate, FOOD=Italian\}, meaning that at \( t_i \) the system assumes that the user is looking for an Italian restaurant with a moderate price.

After being collected through Wizard of Oz, turns of each dialogue are annotated with the corresponding dialogue state, consisting of an intent and a set of slot-value pairs. The following is an example of the annotation provided in a portion of a MultiWOZ 2.0 dialogue:

**USER:** I would like a moderately priced restaurant in the west part of town.
**SYSTEM:** There are three moderately priced restaurants in the west part of town. Do you prefer Indian, Italian or British?
**USER:** Can I have the address and phone number of the Italian location?

3 Dialogue Domain Adaptation: Task Definition

In this section we define dialogue domain adaptation (DDA) and its core properties. In
our setting we assume an initial conversational domain, represented in a KB-SOURCE, and corresponding annotated training dialogues D-SOURCE. Then, as in real application scenarios, we assume that a number of changes occur in KB-SOURCE, such that a new conversational domain KB-TARGET needs to be considered. Dialogue domain adaptation consists in the capacity to automatically produce new annotated dialogues D-TARGET, such that they maintain both the linguistic structure and the linguistic variability of the initial D-SOURCE dialogues, while, at the same time, being consistent with the new KB-TARGET.

Figure 1 provides a concrete example of DDA. Here we have a source dialogue, taken from the MultiWOZ data-set of restaurant booking conversations, mentioning restaurants in a certain region. Notice that it is implicitly assumed that the system responses are true facts in a KB-SOURCE (e.g., if the system says What about Eraina?, it means that KB-SOURCE contains a restaurant named Eraina). However, this might not be true in KB-TARGET, where the Eraina restaurant might not exist anymore. The target dialogue in Figure 1 is basically the same dialogue as the source dialogue, although adapted to be consistent with a KB-TARGET. DDA focuses on the automatic generation of such D-TARGET dialogues starting from D-SOURCE dialogues.

In the rest of the section we consider three core characteristics that affect DDA: domain changes, dialogue internal coherence, and KB-dialogue adherence.

3.1 Domain Changes

DDA strongly depends on the amount and types of changes that differentiate KB-SOURCE from KB-TARGET. Intuitively, the more the changes, the more the difficulty to adapt D-SOURCE to dialogues consistent to KB-TARGET.

As described in Section 2, we assume that domain knowledge is represented through a domain ontology, providing a schema that describes entities with slots and corresponding slot-values, and through a knowledge base, providing instances of domain entities. Accordingly, changes in domain knowledge may occur in four cases: (i) a domain is introduced or removed (e.g., adding HOTEL); (ii) a slot for a domain is introduced or removed (e.g., adding PARKING among the slots for RESTAURANT); (iii) a slot-value for an existing slot is introduced or removed (e.g., adding TUSCAN as slot-value for the slot FOOD of the RESTAURANT domain); (iv) an instance for a certain domain is introduced or removed (e.g., adding a new restaurant like BELLA NAPOLI with its features). In a concrete situation, such changes may occur either in an incremental way, as small changes of the domain knowledge reflecting modifications of the world, or as a consequence of a domain shift, when, for instance, a dialogue system is moved from one city to another.

In this paper we focus on slot-value changes, and we assume that, while moving from KB-SOURCE to KB-TARGET, both domains, slot names and number of instances are kept without any change.
3.2 Dialogue Internal Coherence

Human collected dialogues (as D-SOURCE dialogues) possess an internal coherence that needs to be preserved in D-TARGET dialogues. As an example, in the source dialogue in Figure 1, we assume that the Eraina restaurant mentioned by the system is coherent with the request of the user for an *European* restaurant. We assume that co-reference between anaphoric expressions and their references (e.g., *those* on the D-SOURCE) are kept consistent within the scope of a dialogue. Similarly, language variations (e.g., using different spellings for referencing the same entity) should be used consistently in the same dialogue.

Moreover, the semantic annotations of the dialogues need to respect the references of the utterance, even when anaphoric expressions occur. For example, if the user says *I want to book a table on the same day as my train arrival*, the annotation for "booking-day" has to be consistent to the referent mentioned in the previous part of the conversation.

3.3 KB-Dialogue Adherence

The core assumption behind *dialogue domain adaptation* is that system utterances have to be as much as possible aligned with domain knowledge, meaning that the system responses should correspond to true facts in the domain knowledge. As an example, in Figure 1, the D-SOURCE dialogue reports that there are 5 restaurants in KB-SOURCE with certain characteristics, while the corresponding D-TARGET turn has been adapted reporting 4 restaurants in the KB-TARGET.

When the dialogue collection is carried on manually, KB-Dialogue adherence is supposed to be checked by humans, so that each system utterance is coherent to the KB. However, human mistakes may occur, for instance in case crowd workers in a Wizard of OZ setting make wrong queries to the domain KB. The relevance of KB-Dialogue adherence in our experimental setting will be discussed in Section 5.

4 Substitution-based DDA

We approach the dialog domain adaptation task described in section 4 through the substitution of slot-values in D-SOURCE with slot-values selected from KB-TARGET. Figure 2 depicts the elements of our experimental setting, highlighting the relationships between them.

The dataset D-SOURCE consists of both training and test dialogues, with the latter possibly containing a certain number of slot-values that are unseen in the training set. D-SOURCE dialogues are collected in a strong connection with KB-SOURCE, which has been quantified through a KB-Dialogue adherence measure. Given a certain KB-TARGET, which differs from KB-SOURCE for a proportion of slot-values estimated by the KB-overlap, the KB mapping defines the substitutions that need to be done for every slot-value, and, on the basis of this mapping, the adaptation process generates the D-TARGET data-set.

![Figure 2: Scheme of dialogue domain adaptation methodology.](image)

4.1 D-TARGET Generation

Starting from an annotated D-SOURCE dialogue, a KB-SOURCE and a KB-TARGET, the creation of a D-TARGET dialogue follows a general substitution-based procedure. For each slot-value found in D-SOURCE, according to the semantic annotations provided in the dataset, the first step consists of checking whether the slot-value is known in KB-SOURCE. If it is known, then we try to substitute it with a corresponding slot-value in the KB-TARGET. Otherwise, we keep it as it is in D-TARGET.

In order to check whether the slot-value is known in KB-SOURCE, we compare the source slot-value with all slot-values in KB-SOURCE that have the same slot-name, applying a similarity function based on a variation of the Gestalt Pattern Matching algorithm (Black, 2004). We used a
threshold for deciding when to consider two slot-values as equal or different. The threshold has been determined by using a data-set of positive and negative examples and empirically finding the value that best separates the two sets (e.g., with the similarity value of 0.6, 10 slot-value pairs are classified as equal while they are different, and 8 pairs are classified as different while they are the same).

If at least one slot-value in KB-TARGET with similarity above the threshold is found, then we apply a mapping function that selects a target slot-value \( v_t \) to be used for substitution. We have defined three mapping functions.

**Random-DIALOGUE.** For every slot-value \( v_i \) in a D-SOURCE dialogue, this mapping function randomly chooses one value from all the slot-values in the dialogue, both from User and System, that have the same slot-name as \( v_i \). Then, the randomly chosen slot value is assigned to \( v_t \).

**Random-KB.** For every slot-value \( v_i \) in KB-SOURCE, this mapping function randomly chooses one value from all the slot-values in the KB-TARGET that have the same slot-name as \( v_i \). Then, the randomly chosen slot value is assigned to \( v_t \).

**Frequency-KB.** For every slot-value \( v_i \) in KB-SOURCE, this mapping function chooses a value in KB-TARGET that has the same slot-name as \( v_i \), on a frequency mapping (e.g., “indian” may correspond to “italian” if the proportion of instances for the two slot-value is similar). Then, the chosen slot value is assigned to \( v_t \).

In order to generate a D-TARGET dialogue, we then go through all the slot-values in D-SOURCE and for each of them we check if there is a \( v_t \) mapping. If a mapping is found, we perform the substitution, otherwise we leave it as it is.

### 4.2 KB Overlap
We use *Knowledge Bases Overlap* as a measure that determines how much KB-SOURCE is equivalent to KB-TARGET. In order to assess this, all unique possible values for every slot of the domains in one KB need to be compared to all the unique values of the same slot for the other KB. For instance, if the slot is "restaurant-area", KB-SOURCE may have values ["north", "south", "east", "west"], while KB-TARGET may have values ["centre", "north", "south"]. In such case, the equal values for the slot would be 2, and the total different values would be 5, resulting in a KB overlap of 40%. In other words, the KB overlap indicates the percentage of changes that need to be done for changing from one KB to another.

#### 4.3 Estimating KB-DIALOGUE Adherence
In accordance to what has been defined in paragraph 3.3, we intend the KB-DIALOGUE Adherence as the extent by which a dialogue is consistent to the content of the KB. In order to estimate this, we distinguish two cases:

- Case 1: the slot-values of one instance mentioned in the utterance correspond to the description of the instance in the KB (e.g., "The Old Cambridge is an expensive restaurant in the centre").
- Case 2: the system states a certain number of instances that meet certain conditions, and the KB actually contains the same number of instances (e.g. "There are 15 hotels with 4 stars in the north").

The adherence for each case is given by the percentage of the system’s utterances that complies with the respective condition, and the total KB-dialogue adherence is then calculated by averaging the two cases.

#### 4.4 Unseen Slot-value Ratio
Given a dialogue data-set split into training and test set, the *unseen slot-value* ratio measures the number of slot-values that are present in the test set dialogues, but that are not present in the training set dialogues. This is an important indicator, which significantly affects the performance of a model, as, for every unseen slot-value, the model has to make a prediction over something for which it had not been trained on.

### 5 Experimental Setting
This section describes the experiments that we carried on to test *dialogue domain adaptation* based on slot-value substitutions. The experimental setting includes an initial KB-SOURCE and corresponding D-SOURCE dialogues; a set of handcrafted test D-SOURCE dialogues; few substitution algorithms that we experimented to produce different D-TARGET dialogues; and a state-of-art dialogue state
tracker to check the performance of different adaptation strategies.

5.1 KB-SOURCE and D-SOURCE

Our experimental setting is derived from the MultiWOZ 2.3 data-set (Han et al., 2020). Experimental D-SOURCE dialogues consists of the 10,438 MultiWOZ dialogues, with an average of 11.06 turns per dialogue, collected with the technique of the Wizard of Oz and spanning over 7 domains: Train, Attraction, Restaurant, Hotel, Police, Taxi, Hospital. Through dialogues the user asks information about things to do in Cambridge, such as restaurant or hotel reservation, request for train timing, information about an attraction, etc. The MultiWOZ knowledge base (i.e., our KB-SOURCE) presents an average of 525 instances per domain and 8.5 slots per instance.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Slot Type</th>
<th>Slot-value union</th>
<th>Intersection</th>
<th>Overlap %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attrac.</td>
<td>inf.</td>
<td>24</td>
<td>20</td>
<td>83.33</td>
</tr>
<tr>
<td></td>
<td>all</td>
<td>862</td>
<td>139</td>
<td>16.13</td>
</tr>
<tr>
<td>Hosp.</td>
<td>inf.</td>
<td>118</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>all</td>
<td>239</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>Hotel</td>
<td>inf.</td>
<td>18</td>
<td>18</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>all</td>
<td>361</td>
<td>83</td>
<td>22.99</td>
</tr>
<tr>
<td>Police</td>
<td>inf.</td>
<td>2</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>all</td>
<td>7</td>
<td>1</td>
<td>14.29</td>
</tr>
<tr>
<td>Rest.</td>
<td>inf.</td>
<td>47</td>
<td>15</td>
<td>31.91</td>
</tr>
<tr>
<td></td>
<td>all</td>
<td>1173</td>
<td>129</td>
<td>11.00</td>
</tr>
<tr>
<td>Train</td>
<td>inf.</td>
<td>1226</td>
<td>184</td>
<td>15.01</td>
</tr>
<tr>
<td></td>
<td>all</td>
<td>5703</td>
<td>699</td>
<td>12.26</td>
</tr>
<tr>
<td>All by slots</td>
<td>inf.</td>
<td>1435</td>
<td>237</td>
<td>16.52</td>
</tr>
<tr>
<td></td>
<td>all</td>
<td>8345</td>
<td>1051</td>
<td>12.59</td>
</tr>
<tr>
<td>All by domains</td>
<td>inf.</td>
<td>2846</td>
<td>454</td>
<td>38.38</td>
</tr>
<tr>
<td></td>
<td>all</td>
<td>15828</td>
<td>1963</td>
<td>12.78</td>
</tr>
</tbody>
</table>

Figure 3: Slot-value overlap between Cambridge KB-SOURCE and Pisa KB-TARGET.

5.2 KB-TARGET

We decided to experiment DDA on a conversational domain with similar characteristics as MultiWOZ, simulating an application for the city of Pisa, in Italy. Pisa presents a number of characteristics that are very similar to Cambridge, such as the dimension, the presence of an important University with many departments spread all over the city, and the characterization of being a touristic city. The information necessary to create the Pisa KB-TARGET has been taken from a number of publicly available data-sets1. Starting from the MultiWOZ KB, as discussed in Section 3, we focused on slot-value changes, i.e., preserving all information but slot-values. The overlap between the resulting Pisa KB-TARGET and the initial Cambridge KB-SOURCE is shown in Table 3. We show both the breakdown slot-value overlap for single domains, as well as the aggregate overlap by slots and domains. Overall, 12.59% of the slot-values overlaps, indicating that the domain shift from Cambridge to Pisa has produced a drastic change in term of slot-values.

5.3 Test D-TARGET Dialogues

We created a Pisa test set (Pisa-T), using the test portion of the FREQUENCY KB Pisa data-set, which has then been manually revised with the aim of creating an error free data-set with respect to the Pisa KB, for what regards system messages. This means that every system utterance should tell the truth relatively to what is contained in the KB.

Test dialogues have been produced according to a semi-automatic procedure. First we apply substitutions to the original MultiWOZ test dialogues according to the frequency strategy described in section 4.1. In order to perform these corrections, we automatically identified the dialogues that showed a lack of adherence, and we adjusted them manually, both for the system utterances and for the semantic annotations, making sure that all test dialogues comply with the information reported in the respective KBs.

5.4 DDA Substitution Algorithms

We intend to compare different DDA substitution algorithms against the no-adaptation situation (i.e., the original MultiWOZ). We created three different training datasets that have been used for running experiments against the manually constructed test set.

1http://www.datiopen.it
No Adaptation Cambridge (NO ADAPT.). This is the original dataset from MultiWOZ 2.3. It is the baseline for our experiments, as no dialogue adaptation has been applied.

Random Selection from Dialogues (RANDOM-D). This substitution strategy is intended to preserve as much as possible the linguistic variety of D-SOURCE in D-TARGET. The dataset has been created in two steps: first, a preliminary Pisa dataset has been created with a frequency strategy, then all the slot-values in the dialogue have been randomly shuffled.

Random Selection from KB (RANDOM-KB). This substitution strategy is intended to take advantage of the alignment between KB-SOURCE and KB-TARGET, although with a basic random selection of target slot-values. This strategy does not preserve linguistic variety in D-SOURCE. Starting from the Cambridge dialogues, the substitutions have been done taking one random slot-value from KB-TARGET, only when the original slot-value was present in the KB-SOURCE.

Frequency-based Selection from KB (FREQ-KB). This substitution strategy is intended to take full advantage of the alignment between KB-SOURCE and KB-TARGET, choosing target slot-values that maximise their frequency in KB-SOURCE. This strategy does not preserve linguistic variety in D-SOURCE. Starting from the Cambridge dialogues, the substitutions have been done taking one slot-value, decided on the basis of a frequency strategy, from the KB-TARGET, only when the original slot-value was present in the KB-SOURCE.

### 5.5 DST Model and Evaluation Metrics

We compare the substitution strategies presented in Section 5.4 according to their capacity to provide training data for a dialogue state tracker. For all of our experiments we used the dialogue state tracking algorithm TRADE (Wu et al., 2019a). The algorithm is optimized for being used on multi-domains datasets like MultiWOZ, and it has actually been evaluated on this data-set (in its first version) for assessing the performance during the development of the algorithm.

The main evaluation metric is joint goal accuracy, largely used for DST, defined as the set of accumulated turn level goals up to a given turn in predicting all slots in a given turn correctly and it is computed by the fraction of turns in a dialogue where all slots in a turn are predicted correctly.

### 6 Results and Discussion

Table 1 shows a summary of the results that we obtained from our experiments. We started from the original MultiWOZ 2.3 data-set, referred as "Cam" in the Table, based on the Knowledge Base "Cam-KB". We obtained a Joint Goal Accuracy of 0.490 for "Cam", which is aligned with the value reported for TRADE on MultiWOZ 2.3 (Wu et al., 2019b).

The NO ADAPT. experiment aimed at reproducing a zero adaptation situation, training a model on D-SOURCE and testing it on D-TARGET, which in our case was based on a KB-TARGET that differs of around 88% from the KB-SOURCE (see paragraph 4.2). As expected, the value for the unseen slots is much higher (more than 12 times) compared to the original setting. This contributed to a decrease of almost 75% in the Joint Goal Accuracy performance.

The remaining experiments were ran on three different D-TARGET datasets, created on the basis of different strategies, as explained in Section 5.4. RANDOM-D was made by substituting every source slot-value with a slot-value that was taken randomly from a list of all unique slot-values that are present in the dialogue. This means that for every substitution, there was the same chance of picking a very frequent value - such as "Indian food" - than picking a value that occurs only once - such as "south Caribbean spicy food". For this reason the Joint Goal Accuracy is very low, even if significantly better than in the no adaptation setting. A major improvement, however, happens when we use data-sets that are based on an adapted strategies based on the KB.

The RANDOM-KB strategy, in fact, has a Joint Goal Accuracy slightly lower than the original Cam experiment, while the frequency-based strategy even exceeds Cam with an goal accuracy over 50%. The difference in performance between these last two data-sets can be explained by considering that with RANDOM-KB we randomly changed the assignment of the slot-values to be substituted once for every dialogue, which can be beneficial for the model capability of generalizing, but that does not allow it to maximize the

Table 1: Results of the dialogue domain adaptation experiments. All experiments use the TRADE model. The first row corresponds to the original MultiWOZ 2.3 dataset tested over itself. The second row is the same dataset tested over our Pisa dataset, which has been manually ensured to be perfectly fitted to the KB-TARGET, and which has also been used for testing the other adaptation strategies.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Training</th>
<th>Test</th>
<th>KB</th>
<th>Unseen slot-values</th>
<th>Joint Accuracy</th>
<th>KB train adherence</th>
<th>KB test adherence</th>
</tr>
</thead>
<tbody>
<tr>
<td>——</td>
<td>Cam</td>
<td>Cam-T</td>
<td>Cam-KB</td>
<td>1.22%</td>
<td>0.490</td>
<td>87.99%</td>
<td>91.66%</td>
</tr>
<tr>
<td>NO ADAPT.</td>
<td>Cam</td>
<td>Pisa-T</td>
<td>Pisa-KB</td>
<td>15.19%</td>
<td>0.131</td>
<td>39.22%</td>
<td>100%</td>
</tr>
<tr>
<td>RANDOM-D</td>
<td>Pisa</td>
<td>Pisa-T</td>
<td>Pisa-KB</td>
<td>2.27%</td>
<td>0.239</td>
<td>42.91%</td>
<td>100%</td>
</tr>
<tr>
<td>RANDOM-KB</td>
<td>Pisa</td>
<td>Pisa-T</td>
<td>Pisa-KB</td>
<td>4.78%</td>
<td>0.461</td>
<td>39.25%</td>
<td>100%</td>
</tr>
<tr>
<td>FREQ. KB</td>
<td>Pisa</td>
<td>Pisa-T</td>
<td>Pisa-KB</td>
<td>4.8%</td>
<td>0.502</td>
<td>83.96%</td>
<td>100%</td>
</tr>
</tbody>
</table>

6.1 Linguistic Variability

The different substitutions approaches that we have adopted, radically diverge in the linguistic variability they produce in D-TARGET. On one side, for the RANDOM-D dialogues we substitute slot-values that are taken from the dialogue, this way preserving their variability (e.g., typos, synonyms, abbreviations, etc.), on the other side, for the RANDOM-KB and FREQUENCY KB, we always substitute slot-values that are present in the KB, which are only in their normalized version. This way we flatten the linguistic variability, and significantly reduce the total number of unique slot-values in the dialogue. This aspect is clearly highlighted by the different values for the Unseen Ratio. While RANDOM-D, which has a high number of variations for every slot-value (even if not as high as in Cam), produces a percentage of 2.27, RANDOM-KB and FREQUENCY KB show an Unseen Ratio percentage of more than the double. By substituting all possible variations of a slot-value with one unique value, in fact, the portion of slot-values that are not seen in the training, but that instead are present in the test, strongly increases.

7 Conclusion

Domain knowledge in conversational agents is subject to frequent changes, and this leads to the necessity of continuously updating training dialogues in order to keep them consistent with domain knowledge. As collecting conversational dialogues by hand requires a significant effort, approaches for automatically updating are required. In this paper we have proposed dialogue domain adaptation, a methodology for operating changes to an initial training dialogue, so that it becomes adherent to a modified domain knowledge. The experiments that we conducted reveal a twofold evidence: they demonstrate that zero adaptation results in a significant loss in DST performance, and they show that simple substitution-based adaptation methods bring instead effective results. Moreover, the experiments on different adaptation methods showed diverse phenomena. While the best performance is obtained using a frequency strategy - which maps the most frequent slot-value of the source domain to the most frequent slot-value of the target domain - a random strategy based on KB values performed slightly worse, and a severe drop in the results occurred when using a random strategy based on dialogue values. Linguistic variability is perhaps an important factor that emphasises this difference, and it will be an interesting topic to be explored in further works.
References


Abstract

The use of pretrained language models, fine-tuned to perform a specific downstream task, has become widespread in NLP. Using a generic language model in specialized domains may, however, be sub-optimal due to differences in language use and vocabulary. In this paper, it is investigated whether an existing, generic language model for Swedish can be improved for the clinical domain through continued pretraining with clinical text.

The generic and domain-specific language models are fine-tuned and evaluated on three representative clinical NLP tasks: (i) identifying protected health information, (ii) assigning ICD-10 diagnosis codes to discharge summaries, and (iii) sentence-level uncertainty prediction. The results show that continued pretraining on in-domain data leads to improved performance on all three downstream tasks, indicating that there is a potential added value of domain-specific language models for clinical NLP.

1 Introduction

Pretrained language models, trained on a variety of readily accessible and large-scale unlabeled corpora, and subsequently fine-tuned on downstream tasks using labeled datasets, have led to substantial performance gains across a whole host of NLP tasks. This has contributed to ameliorating a major bottleneck in the development of NLP systems, i.e. the need for access to very large amounts of labeled data for supervised learning.

In many cases, obtaining large, task-specific datasets in the form of human-annotated corpora is challenging and prohibitively expensive. As a result, the paradigm of pretraining and fine-tuning has become fundamental for contemporary NLP. In particular, with the introduction of models such as BERT (Devlin et al., 2018), which are based exclusively on self-attention, i.e. Transformers (Vaswani et al., 2017), and leverage transfer learning techniques, language models have become increasingly accessible; yet, pretraining language models from scratch requires substantial computational resources.

While generic language models, trained and released to the public by resource-rich organizations, can be utilized and fine-tuned to perform a particular downstream NLP task without a need for significant resources – neither computational nor in terms of data – it has been shown that their use in specialized domains may be sub-optimal as a result of differences in, for instance, language use and vocabulary (Lewis et al., 2020; Gururangan et al., 2020). This has motivated efforts to develop domain-specific language models, e.g. SciBERT (Beltagy et al., 2019) and BioBERT (Lee et al., 2020).

Specialized language models have been developed either by (i) pretraining a language model with in-domain data from scratch, possibly in combination with out-domain data, or by (ii) continuing to pretrain an existing, general language model with in-domain data (domain-adaptive pretraining), either by using large amounts of in-domain data, if available, or by only using task-related unlabeled data (task-adaptive pretraining).

The need for language models is particularly pronounced in low-resource settings – both in terms of languages and domains. While there is a publicly available generic language model for Swedish, KB-BERT (Malmsten et al., 2020), pretrained using text from the National Library of Sweden, there is no domain-specific variant for Swedish clinical text.

In this paper, we report on the development of a clinical language model for Swedish. The approach is based on continual pretraining of KB-BERT with in-domain data in the form of clinical text. The
model is fine-tuned and evaluated on three downstream clinical NLP tasks: (i) detection of protected health information, i.e. a named entity recognition task, (ii) automatic assignment of ICD-10 codes to discharge summaries, i.e. a document-level multiclass, multi-label classification task, and (iii) uncertainty classification, i.e. a sentence-level multiclass, single-label classification task. The clinical KB-BERT is compared to the original KB-BERT and we report downstream performance of various checkpoints during the pretraining process. The domain-specific and generic BERT models are further evaluated on a generic NER task in order to understand if performance gains are best explained by the quantity or the domain-specificity of the additional pretraining data.

2 Related Work

There has been a substantial amount of effort in recent years dedicated to exploring and developing domain-specific and specialized language models by pretraining with in-domain data, particularly for the biomedical domain and for English.

An early and notable effort was the release of BioBERT (Lee et al., 2020), a BERT model pretrained on large-scale biomedical corpora (PubMed abstracts and PMC full-text articles) in addition to general-domain corpora (English Wikipedia and BooksCorpus). Rather than training the model from scratch, BioBERT was initialized with the general-purpose BERT model and also inherited its vocabulary, after which pretraining continued using biomedical data. It was shown that BioBERT significantly outperforms BERT on biomedical NLP tasks.

Subsequent efforts, e.g. BioMegatron (Shin et al., 2020), have shown that additional improvements can be gained by training larger models on even larger in-domain corpora and, in some cases, using a domain-specific vocabulary. In another study, experimental results indicated that training biomedical language models from scratch, as opposed to continued pretraining of a generic language model, may yield improved performance on downstream domain-specific tasks (Gu et al., 2020), although requiring substantial computational resources.

Domain-specific language models have also been developed for the clinical domain, albeit not for Swedish. Alsentzer et al. (2019) pretrained clinical BERT models on MIMIC-III (Johnson et al., 2016) using either (i) all types of clinical notes or (ii) discharge summaries only.

It was found that initializing the clinical BERT models with parameters from BioBERT, as opposed to parameters from BERT, led to better downstream performance, while the types of clinical notes used made little difference on most downstream tasks. However, the clinical language models yielded an increased performance on some – but not all – of the clinical NLP tasks compared to BERT and BioBERT.

There have been efforts to develop language models using a combination of biomedical and clinical data. In Lewis et al. (2020), the authors develop such models by applying recent advances in pretraining introduced by RoBERTa (Liu et al., 2019), while studying the impact of using different (combinations of) training corpora and model sizes along with a domain-specific vocabulary. Liu et al. (2019) compare their models with previously published language models on a number of downstream tasks in different domains. Their results suggest that using a larger, more powerful general-purpose language model may be better than using a smaller, less powerful domain-specific language model. However, it is also shown that using in-domain data does lead to improved performance: in particular, using clinical data for pretraining leads to large performance gains on clinical tasks but has little impact on biomedical tasks. Learning a domain-specific vocabulary yielded improvements on sequence labeling tasks, while the impact was less clear for classification tasks.

Another very relevant study was conducted by Gururangan et al. (2020), where they also explore the potential advantages in continuing to pretrain an existing BERT model with in-domain data. The authors explore a number of different settings, such as continuing pretraining on a collection of in-domain corpora for a limited amount of time, continuing the pretraining on the unlabeled training set of the intended downstream task, or continuing pretraining on available unlabeled data directly related to the future downstream task at hand. They explore the duration of each of the continued pretraining setups, and they show that this approach can be very beneficial, especially for the setup in which unlabeled data related to the task at hand is exploited for further pretraining. These results are promising and partly inspired the current work, as all the annotated corpora are very small subsets of
the pretraining data.

3 Methods and Data

In this paper, we report on the development of a clinical language model for Swedish. The hypothesis is that a clinical language model, in this case obtained by continuing to pretrain KB-BERT (Malmsten et al., 2020) – a generic language model for Swedish – using large amounts of in-domain clinical text, will yield improved performance over a generic language model on downstream clinical NLP tasks. The pretraining process of KB-BERT is continued using 17.8 GB of clinical text from the research infrastructure Health Bank – Swedish Health Record Research Bank at DSV/Stockholm University (Dalianis et al., 2015). This is, in fact, a similar amount of data that was used for pretraining KB-BERT.

The clinical BERT model is pretrained using a GeForce RTX 1080 GPU and 17.8 GB of uncompressed clinical text in the form of all available types of clinical notes over a period of seven years. The clinical BERT model is trained for one epoch, corresponding to a total duration of ten days. The clinical BERT model is fine-tuned and evaluated on three downstream clinical NLP tasks: (i) detection of protected health information, i.e. a named entity recognition task, (ii) automatic assignment of ICD-10 codes to discharge summaries, i.e. a document-level multi-class, multi-label classification task, and (iii) uncertainty classification, i.e. a sentence-level multi-class, single-label classification task. The clinical BERT model is compared to the original KB-BERT and we report downstream performance of various checkpoints during the pretraining process.

Furthermore, as the clinical BERT model is developed using more data compared to KB-BERT, it is important to investigate whether potential differences in downstream performance can be attributed to the amount of pretraining data rather than the domain-specificity of the data. To that end, the two language models are also evaluated on a NER task in the general domain. A similar improvement on this task could imply that the clinical BERT model is primarily benefiting from additional pretraining data, whereas degraded performance would indicate the value of pretraining specifically on in-domain data.

3.1 Data

Health Bank contains over 2 million patient records encompassing 500 clinical units from the years 2007-2014 from Karolinska University Hospital in Sweden. All clinical notes available in Health Bank – comprising 17.8 GB of uncompressed text – are used for continued pretraining of KB-BERT in order to develop a clinical BERT model for Swedish. In addition, the following four annotated datasets are used for fine-tuning and evaluation, corresponding to three clinical NLP tasks and one generic NLP task.

The Stockholm EPR PHI Corpus comprises 21,653 sample sentences, 380,000 tokens and contains 4,480 annotated entities corresponding to 9 PHI (Protected Health Information) classes: First Name, Last Name, Age, Phone Number, Location, Health Care Unit, Organisation, Full Date, and Date Part. Identifying PHI in clinical notes is a fundamental step in de-identification and is typically approached as a NER task. Details about the dataset can be found in (Dalianis and Velupillai, 2010a).

The Stockholm EPR Gastro ICD-10 Corpus, or ICD-10 Corpus for short, consists of 6,062 samples in the form of discharge summaries belonging to a number of ICD-10 diagnosis blocks. This is a document-level multi-class, multi-label classification task, where the ICD-10 codes are grouped into 10 groups with a more coarse granularity compared to the full ICD-10 codes. These groups are decided based on body parts and range from the ICD-10 code K00 to K99 and average 1.2 labels per sample. Details about the dataset can be found in (Remmer, 2021; Remmer et al., 2021).

The Stockholm EPR Sentence Uncertainty Corpus contains 5,515 samples in the form of sentences classified as Certain, Uncertain, and Undefined. The dataset is highly unbalanced with 88% of the samples belonging to the Certain class, 10% to the Uncertain class and the rest to the Undefined class. This is a sentence-level multi-class, single-label classification task. Details about the dataset can be found in (Dalianis and Velupillai, 2010b).

Swedish Web News Corpus comprises approximately 8,000 samples in the form of sentences and contain the entities PER, LOC, ORG and MISC.

---

1This research has been approved by the Swedish Ethical Review Authority under permission no. 2019-05679.
2http://dsv.su.se/healthbank
The dataset comes from Webbnyheter 2012\textsuperscript{4}, which was annotated semi-automatically. This dataset is used for the general-domain NER task.

### 3.2 Pretraining

The pretraining of the model with the Masked Language Modeling (MLM) task started from the released checkpoint of KB-BERT and lasted for 40,000 steps, approximately corresponding to one data epoch, or ten days in real time for our GPU. The pretraining procedure of BERT is closely followed with the notable difference that only sequence lengths of 512 are pretrained, acknowledging the significant evidence in the literature suggesting improved performance in later downstream tasks (Liu et al., 2019). Due to the high variability of the different note lengths in the pretraining data, and to construct the 512 chunks of text, the aforementioned work is carefully followed and each note is treated as a document, concatenating the different notes and separating them with extra [SEP] tokens to indicate the end of each document. Lastly, following the original BERT pretraining, a learning rate of 1 · 10\(^{-4}\) with a linear schedule is used, a batch size of 256 utilizing gradient accumulation, and 10,000 warm up steps. Below, in Table 1, the hyper parameters of the pretraining session are presented.

<table>
<thead>
<tr>
<th>Hyper parameters</th>
<th>KB-BERT</th>
<th>Clinical KB-BERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>learning rate</td>
<td>10(^{-4})</td>
<td>10(^{-4})</td>
</tr>
<tr>
<td>batch size</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>Adam optimizer</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(\beta_1)</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>(\beta_2)</td>
<td>0.999</td>
<td>0.999</td>
</tr>
<tr>
<td>L2 weight decay</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>warm up steps</td>
<td>10,000</td>
<td>10,000</td>
</tr>
<tr>
<td>dropout</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>linear learning rate decay</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>update steps</td>
<td>1,000,000</td>
<td>+40,000</td>
</tr>
<tr>
<td>training sequence length</td>
<td>128 and 512</td>
<td>only 512</td>
</tr>
<tr>
<td>MLM probability</td>
<td>15%</td>
<td>15%</td>
</tr>
</tbody>
</table>

Table 1: Pretraining hyper parameters comparison with original KB-BERT.

### 3.3 Fine-tuning

The primary way in which pretrained models can be evaluated is to fine-tune them to perform a number of tasks and evaluating their performance on these downstream tasks. Utilizing transfer learning, BERT allows for fine-tuning a model to any traditional NLP task with minimal changes. For each task, the core of the language model is kept intact, in this case the KB-BERT model or the subsequent checkpoints of clinical KB-BERT, and only the final classification layers are changed as appropriate depending on the task. The parameters of BERT are not held frozen but are updated for each task since this has been shown to yield an increased performance compared to only training the final layer (Devlin et al., 2018). As the main aim is to compare the further pretrained checkpoints of KB-BERT with the original KB-BERT, an extensive hyper-parameter search is avoided for the different downstream tasks; instead, the hyper parameters used are within the suggested ranges described by Devlin et al. (2018). As such, and as shown in Table 2, a batch size of 32 is used with a learning rate of 2 · 10\(^{-5}\) for the multi-label classification task, while a batch size of 64 along with a learning rate of 3 · 10\(^{-5}\) is used for the NER and multi-class classification tasks. In all of the cases, training proceeds until loss convergence and early stopping is utilized to stop the training process at that point.

<table>
<thead>
<tr>
<th>hyper parameters</th>
<th>PHI</th>
<th>ICD-10</th>
<th>Uncertainty</th>
<th>Web news</th>
</tr>
</thead>
<tbody>
<tr>
<td>learning rate</td>
<td>3 · 10(^{-4})</td>
<td>2 · 10(^{-4})</td>
<td>3 · 10(^{-4})</td>
<td>3 · 10(^{-4})</td>
</tr>
<tr>
<td>batch size</td>
<td>64</td>
<td>32</td>
<td>64</td>
<td>64</td>
</tr>
</tbody>
</table>

Table 2: Fine-tuning hyper parameters.

However, it should be noted that when the goal is to reach the best possible performance, it is critical to perform a proper hyper-parameter search, as other parameter choices may yield a better result. Furthermore, learning rate schedulers, warm up steps, and gradient constraining approaches, such as gradient clipping, should also be explored as possible performance-enhancing changes. In this study, no extensive hyper-parameter search is conducted, nor are other optimization techniques applied, as the goal is to compare the relative performance of two models rather than obtaining state-of-the-art results on the downstream tasks.

All tasks were performed in a conventional setup where three subsets of each dataset are used: a training set that contains approximately 80% of the dataset, a validation set containing approximately 10%, and a test set containing approximately 10%. However, in the case where a train-test split is already provided, as in the case of the Web News Corpus, the test set is left unchanged and the training-validation split corresponds to 90-10% of the original training set.

\textsuperscript{4}https://spraakbanken.gu.se/en/resources/webbnyheter2012
4 Results

After fine-tuning KB-BERT and clinical KB-BERT to each of the four tasks – three clinical and one generic – they are evaluated and the results are reported in Table 3 below.

<table>
<thead>
<tr>
<th>dataset</th>
<th>model</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHI (Clinical)</td>
<td>KB-BERT</td>
<td>90.53%</td>
<td>90.34%</td>
<td>90.98%</td>
</tr>
<tr>
<td>ICD-10 (Clinical)</td>
<td>KB-BERT</td>
<td>85.53%</td>
<td>75.75%</td>
<td>80.35%</td>
</tr>
<tr>
<td>Uncertainty (Clinical)</td>
<td>KB-BERT</td>
<td>86.83%</td>
<td>79.06%</td>
<td>82.76%</td>
</tr>
<tr>
<td>Web news (General)</td>
<td>KB-BERT</td>
<td>89.81%</td>
<td>82.47%</td>
<td>84.14%</td>
</tr>
</tbody>
</table>

Table 3: Comparison of the performance of KB-BERT with its clinical KB-BERT counterpart after the end of 1 epoch of further pretraining on 17.8 GB of clinical text. The first three tasks belong to the clinical domain, while the fourth task belongs to the general domain. The best scores are annotated in bold.

The results indicate that there is a clear benefit in continuing the pretraining process with in-domain data. The clinical KB-BERT outperforms its KB-BERT counterpart on all three clinical NLP tasks, in terms of F1-score. On the PHI NER task, it performs close to two percentage points better in terms of F1-score, with a large increase in recall and more or less the same precision. The improvement on the ICD-10 code assignment task is in the same range as the PHI NER task, but in this case yielding a further increase of both precision and recall. On the Uncertainty task, the performance improvement is not quite as large as for the other two clinical tasks. However, clinical KB-BERT still improves in all the metrics when compared to its KB-BERT counterpart.

However, on the general-domain NER task, the clinical KB-BERT underperforms compared to KB-BERT. It falls short by around 4 percentage points in terms of F1-score and recall, and by more than 2 percentage points in terms of precision. This indicates that adding more pretraining data does not necessarily lead to better downstream performance, and also that the improved performance on the clinical NLP tasks can likely be attributed to including in-domain data specifically, and not simply more data in general.

Furthermore, a number of checkpoints during the pretraining process of clinical KB-BERT are evaluated on the downstream tasks, the results of which are reported in Figure 1. As can be seen in the figure, for the clinical NLP tasks, there is a positive trend in the performance as the pretraining session progresses. This indicates that, as more data is used, clinical KB-BERT becomes better at incorporating and encoding the differences in the distribution of the clinical text and, as a consequence, it becomes better at performing the downstream tasks.

However, in the case of the Uncertainty multiclass classification task, this trend is not quite as clear: although the vast majority of checkpoints of clinical KB-BERT seem to benefit from the continued pretraining with in-domain data, it experiences a low spike towards the end of the epoch, recovering right at the end. In contrast, the performance of clinical KB-BERT in the general-domain downstream task seems to follow a steadily degrading trend as the pretraining epoch progresses, and does not show any clear signs of recovering.

Finally, to illustrate the differences between a general-domain corpus and a clinical-domain corpus, the KB-BERT tokenizer is used to process the texts in the PHI Corpus and the Web News Corpus, respectively. This tokenizer is a word piece tokenizer, as described by Schuster and Nakajima (2012), and is responsible for constructing the vocabulary – gradually building it from the character level and upwards – by maximizing the likelihood of the training data with respect to the vocabulary. The goal is to investigate how the sentences in the general-domain and clinical-domain corpora are split into tokens, subtokens, and character-level tokens. This is done by calculating the average sentence length, in terms of number of tokens, in the respective corpora when applying the KB-BERT tokenizer versus tokenization based on whitespace and regular expressions. As demonstrated in Figure 2, the clinical-domain corpus, after being preprocessed with the KB-BERT wordpiece tokenizer, leads to a larger increase in average sentence length compared to the general-domain corpus.

5 Discussion

As demonstrated by the experimental results, there is potentially much to be gained from continuing the pretraining process of an existing generic language model with in-domain data, confirming the findings of previous work. Adapting a generic language model to a specific domain by exploiting the availability of unlabeled in-domain data helps BERT to better capture the semantics of the target domain as reflected by differences in the underly-
ing distribution. This is highlighted, not only by improved performance on the clinical tasks, but also by the decreased performance on the general-domain task. These results allow us to better understand the reasons behind the improved performance in the target domain and is an aspect that is often overlooked in similar studies. It indicates that improvements yielded by clinical KB-BERT are not solely due to being pretrained on more data – irrespective of domain – but that the domain of the data used for pretraining is indeed an important factor.

It is interesting to observe that these improvements were yielded by continuing to pretrain the language model for only one epoch, and it is possible that further improvements could be obtained by continuing to pretrain on the in-domain data for several more epochs. Moreover, in contrast to similar studies, we also evaluate numerous checkpoints during the pretraining of the the clinical language model. An important observation is that the clinical KB-BERT outperforms the original KB-BERT on all three clinical NLP tasks after using only 20% of the in-domain data. This indicates that it may be worthwhile to adapt general language models and
make them domain-specific even in the absence of enormous amounts of in-domain data.

The general domain and the clinical domain differ primarily in the use of different vocabularies. The vocabulary of a newspaper article, or a work of literature, follows a different language distribution compared to a clinical note or a discharge summary. Clinical texts are typically written in a rather peculiar fashion and contain a large amount of technical terms, as well as (ad-hoc) acronyms and abbreviations, that are not as prevalent and may not even exist in the general domain. There may also be domain-conflicting homonyms, where a word has a completely different meaning in one domain compared to another. Due to these differences in vocabulary and frequency, the result of applying a generic language model’s tokenizer – in this case that of KB-BERT – to a clinical corpus is that the words are likely to be split into subwords, even potentially reaching a character-level split. This was indeed confirmed by the analysis presented in Figure 2. This, in turn, entails that the BERT model will use more relevant word-level token representations and more common subword token combinations for the general-domain corpus compared to the clinical-domain corpora, where, instead, there is likely to be a high contribution of subword or even character-level token representations. This impact of the tokenizer in turn implies that the major workload and information encoding falls onto this subset of subword and character-level representations during continued pretraining on in-domain data. This not only helps to explain the increased performance on the clinical tasks, but also potentially the performance degradation on the general-domain task since there is a potential mismatch between the representations that are more frequent in the general domain versus the ones that are more frequent – and updated during continued pretraining – in the clinical domain.

In future work, this challenge regarding tokenization can be addressed by pretraining a clinical language model from scratch, which would create a tokenizer and vocabulary based on the in-domain clinical data. As shown by previous work, this may lead to further improvements in performance on the clinical tasks. Another approach is to manually add specific tokens to the vocabulary of a pretrained model, as explored by Tai et al. (2020). An informed set of tokens could potentially be extracted by a new tokenizer specifically trained on the in-domain data, and in a later step, incorporate the set difference to the original tokenizer’s vocabulary.

Furthermore, we plan to continue pre-training the current clinical BERT model for more epochs in order to investigate whether this will lead to further improvements in performance, as well as training a new model with pseudonymized data with the aim to make this model publicly available.

Lastly, we also plan to explore and compare different transformer approaches, as well as different pretraining continuation setups, such as using specific parts of the dataset in the spirit of Gururangan et al. (2020). These could include more pretraining continuation setups, such as task-specific pretraining, where the unlabeled training set would be used during the pretraining for more epochs.

6 Conclusions

In this paper, we reported on the development of a clinical language model for Swedish – the first of its kind. The development of the domain-specific BERT model followed the common practice of continuing to train an existing generic language model, KB-BERT, with in-domain data. Compared to previous efforts to develop clinical language models for English, the model was trained using non-pseudonymized clinical data and, in contrast to previously reported results (Alsentzer et al., 2019), yielded improvements also on the de-identification sub-task of identifying protected health information in clinical text.

Furthermore, we carefully investigated the effect of further pretraining an existing language model with in-domain data and evaluated a number of checkpoints during the pretraining process on the downstream tasks. The results showed that continued pretraining with in-domain data yielded improvements on the in-domain tasks, but led to worse performance on a general-domain task, indicating that performance gains on the clinical NLP tasks can indeed be attributed to the domain-specificity rather than the sheer size of the additional pretraining data. Finally, these results further demonstrate the value of developing domain-specific and specialized language models.

Acknowledgments

This work was partially funded by the DataLEASH project and by Region Stockholm through the project Improving Prediction Models for Diagnosis and Prognosis of COVID-19 and Sepsis with NLP.
Text Retrieval for Language Learners: Graded Vocabulary vs. Open Learner Model

John S. Y. Lee, Chak Yan Yeung
Department of Linguistics and Translation
City University of Hong Kong
Hong Kong SAR, China
jsylee@cityu.edu.hk, cyyeung91@gmail.com

Abstract

A text retrieval system for language learning returns reading materials at the appropriate difficulty level for the user. The system typically maintains a learner model on the user’s vocabulary knowledge, and identifies texts that best fit the model. As the user’s language proficiency increases, model updates are necessary to retrieve texts with the corresponding lexical complexity. We investigate an open learner model that allows user modification of its content, and evaluate its effectiveness with respect to the amount of user update effort. We compare this model with the graded approach, in which the system returns texts at the optimal grade. When the user makes at least half of the expected updates to the open learner model, simulation results show that it outperforms the graded approach in retrieving texts that fit user preference for new-word density.

1 Introduction

Since language learning requires extensive extra-curricular reading, learners can benefit from a text retrieval system that helps them identify suitable reading materials from a pool of candidate texts (Brown and Eskenazi, 2004; Miltsakaki, 2009; Lee, 2021). The suitability of a text may depend on multiple factors, including the user’s reading interests (Heilman et al., 2007b), and the degree of matching between its difficulty and the user’s proficiency. This paper investigates the use of an open learner model (OLM) to predict the latter.

By giving users more control over their learning, OLMs have been shown to foster users’ confidence and reflection on their progress (Bull and Kay, 2007). These benefits are especially important for long-term or life-long learning activities (Kay and Kummerfeld, 2019), such as foreign language learning. We evaluate the effectiveness of an editable OLM (Bull and Kay, 2010) — i.e., an OLM that allows the user not only to view but also to modify its content — for text retrieval for language learning. As their vocabulary expands, users can update the OLM so that the system continues to retrieve texts that are lexically challenging to them. Specifically, we address two research questions:

Text retrieval performance (Q1): How accurately can an OLM identify reading materials with the desired density of new vocabulary, as specified by the user?

User update effort (Q2): How frequently does the user need to edit the OLM in order to reap its benefits?

Most previous research in computer-assisted language learning measured text retrieval performance through holistic evaluation (Heilman et al., 2007a), or in terms of users’ overall learning outcomes (Hsu et al., 2013). In answering Q1, we will directly evaluate the density of new vocabulary in the retrieved texts. In addressing Q2, we will further consider how retrieval performance is affected by users’ vocabulary acquisition over time and the amount of user update effort.

The rest of the paper is organized as follows. After a review of previous work (Section 2), we define our text retrieval framework (Section 3). We then describe and motivate the user simulation (Section 4). Next, we present the OLM approach and the baseline graded approach (Section 5). Finally, we compare the performance of the OLM and the graded approach with respect to the amount of user update effort (Section 6).

2 Previous work

While text difficulty can be influenced by a variety of lexical, syntactic, semantic and discourse features, many text recommendation systems focus on vocabulary (Brown and Eskenazi, 2004; Hsu et al.,...
2013; Wu, 2016), likely due to the strong correlation between vocabulary difficulty and text difficulty (Heilman et al., 2007b; François and Fairon, 2012). Our study will similarly adopt vocabulary difficulty as the text retrieval criterion.

Various approaches have been proposed for matching language learners with reading materials at a suitable level of lexical difficulty. The graded approach, also known as the leveling approach, places users and documents on a common scale, such as school grades (Mitsakaki and Trout, 2008; Collins-Thompson et al., 2011). The system performs automatic readability assessment on each document and labels it with a grade to reflect its difficulty. This approach may not be able to capture individual learning patterns, however, as users are pigeon-holed into pre-defined grades.

As an alternative, the adaptive approach identifies user traits and preferences, and then adjusts the pedagogical content in the system to optimize learning outcomes (Brusilovsky, 2012; Vandewaetere et al., 2011). In the context of text retrieval for language learning, the system typically maintains a learner model on the user’s linguistic proficiency, and then returns texts that best fit the model. The learner model may be estimated through user updates (Lee, 2021); formal assessment such as cloze items (Heilman et al., 2010); complex word identification models trained on vocabulary self-assessment by the user (Yeung and Lee, 2018); time log and click history patterns (Hokamp et al., 2014); as well as dictionary and translation queries by the user (Wu, 2016), among other non-invasive methods.

One disadvantage of the graded approach is the “jump” in difficulty when promoting the user from one grade to the next. The adaptive approach can potentially provide more fine-grained adjustments by gradually raising the vocabulary difficulty in the retrieved texts. To the best of our knowledge, there has not been any direct, quantitative comparison between these two approaches during a period of vocabulary acquisition by the user. This paper aims to fill in this gap.

3 Text retrieval framework

After motivating the use of vocabulary difficulty as the retrieval criterion (Section 3.1), we describe its implementation in the learner model (Section 3.2) and its application in the retrieval model (Section 3.3).

3.1 Retrieval criterion

The ideal text should have an appropriate amount of new vocabulary, so that it stretches the reader’s competence without hindering comprehension. We quantify vocabulary difficulty by new-word density (Holley, 1973) (NWD), i.e., the percentage of words in the text that are new for the user. This metric is more straightforward to interpret and more transparent than grades, since users can easily examine the basis of retrieval results.

The system aims to return texts at a Target NWD that is specified by the user. The user thus has the freedom to set a relatively high Target NWD, for example, to maximize vocabulary acquisition, or set a relatively low one for leisure reading without dictionary look-ups.

3.2 Learner model

A language learner knows only a limited number of words in the foreign language. For each user $u$, we refer to this set of words as his or her vocabulary set, denoted as $\text{voc}(u) = \{w_1, \ldots, w_n\}$. Although nuances in lexical knowledge may be more precisely expressed with a real-number score (Yimam et al., 2018) or on a Likert scale (Ehara et al., 2012; Shardlow et al., 2021), we opted for the simpler known/unknown distinction to enable an intuitive interpretation of the NWD metric.

Since the system does not know the ground-truth vocabulary set $\text{voc}(u)$, the learner model needs to make an estimation $\hat{\text{voc}}(u)$ for each user $u$. It can be effective to use automatic methods to re-estimate the vocabulary set as the user acquires new vocabulary (Section 2). However, we choose to base our evaluation on manual edits to an open learner model (OLM). This methodology has the advantage of being agnostic to the update algorithm, which may include any combination of manual and automatic methods, and may vary from one text retrieval system to another. Our results will therefore not be tied to any particular algorithm, but rather measure text retrieval performance with respect to varying amounts of valid updates (Section 5.1).

3.3 Retrieval model

The NWD of a document varies according to the user’s vocabulary set. Formally, given a document $d$ with $D$ words, say $d = [w_1, \ldots, w_D]$, its Actual
NWD for user $u$ is:

$$\frac{1}{D} \sum_{i=1}^{D} \text{new}_u(w_i)$$

(1)

where $\text{new}_u(w) = 0$ if the word $w \in \text{voc}(u)$, and $\text{new}_u(w) = 1$ otherwise.

Again, since the system has no access to the ground truth $\text{voc}(u)$, it must use $\text{voc}(\hat{u})$ to compute an Estimated NWD. The retrieval model returns the text whose Estimated NWD is closest to and not exceeding the Target NWD as specified by the user (Section 3.1).

4 Methodology

After motivating the advantages of using a simulation to compare the open learner model (OLM) and the graded vocabulary approach (Section 4.1), we give details on the simulation set-up (Section 4.2) and implementation details (Section 4.3) and define the evaluation metrics (Section 4.4).

4.1 Human subjects vs. user simulation

In the context of this study, text retrieval performance can be influenced by two variables: the Target NWD (Section 3.1) and the frequency of user update to the learner model (Section 3.2). It is therefore helpful to consider multiple configurations of these two variables.

There are a number of trade-offs between a user study and a user simulation. In the former, the subjects would need to perform text searches over a sufficiently long period of time to allow for substantial vocabulary acquisition. Throughout this period, they would need to read the retrieved texts and exhaustively annotate the unknown words therein, while experimenting with various update frequencies. This design has the advantage of providing authentic human data on vocabulary acquisition. However, it would introduce confounding factors such as differences among the subjects’ proficiency levels, ability to work with the user interface, and diligence in updating the learner model. These factors are difficult to control for but can significantly influence the experimental results.

A user simulation can facilitate a more rigorous comparison by keeping these factors constant. It can also cheaply evaluate a large number of text searches, with no constraint on the length of the experimental period. The main disadvantage is that the users’ vocabulary acquisition would need to be prescribed rather than empirically observed.

This issue can be partially mitigated by consulting vocabulary lists, such as the widely used *Hanyu Shuiping Kaoshi* (HSK), which were crafted by experts with support from empirical data to reflect typical language learners (Hanban, 2014).

Given our research goals, we feel that the overall advantages of a simulation outweigh its disadvantages. Our simulation will be able to evaluate over 6K recommended documents in various experimental settings, a set of data points that is an order of magnitude larger than what we would have been able to gather from human subjects.

4.2 Simulation set-up

We simulated a user who searches for extracurricular reading materials for learning Chinese as a foreign language. We ran the simulation three times, with the Target NWD parameter set to $m\%$ NWD, for $m = \{20, 30, 40\}$.

**Text retrieval.** At times $i = 1, \ldots, k$, the user performs a text search to obtain documents whose Estimated NWD is closest to and not exceeding $m\%$ (Section 3.3). The user reads the top-ranked document that he or she has not yet read, and updates the OLM while reading (Section 5.1). Let $d_i$ represent the document read by the user at the $i$th search.

**Vocabulary acquisition.** Between two consecutive searches, the user learns a number of new words. Let $u_i$ represent the user at the $i$th search, and let $W_i$ represent the set of new words learned between the $i$th and $(i + 1)$th searches. The user’s vocabulary set expands during this period as follows:

$$\text{voc}(u_{i+1}) \leftarrow \text{voc}(u_i) \cup W_i$$

(2)

4.3 Simulation implementation

Let $V_l$ denote the accumulative set of words in the HSK graded vocabulary lists up to level $l$, for $l = 1, \ldots, 6$. We set $\text{voc}(u_0) = V_5$ and $\text{voc}(u_6) = V_6$. This means that the user initially knows all the words listed up to HSK level 5, and then learns the words at level 6 during the simulation period.

We set a uniform learning rate at $|W_l| = 6$, meaning that six words are learned between two searches. The acquisition order is in reverse of word frequency in Chinese Wikipedia.

4.4 Evaluation metrics

We use two metrics to evaluate text retrieval performance:
NWD Error The difference between the Actual NWD of \( d_i \) and the Estimated NWD of \( d_i \). This metric measures how much the estimated difficulty of the recommended document deviates from the ground truth. Recall that Estimated NWD is computed according to the estimated vocabulary set \( \text{voc}(\hat{u}_i) \), while Actual NWD is based on \( \text{voc}(u_i) \). These two figures differ whenever new words learned by the user appear in \( d_i \) but have not been updated in the learner model.

NWD Gap The difference between the Actual NWD of \( d_i \) and the Target NWD (Section 3.1). This metric expresses the discrepancy between the actual difficulty of the recommended document and the difficulty requested by the user.

5 Approach

As users learn new words, a document’s new-word density (NWD) decreases. Periodic updates to the vocabulary set and re-estimation of the NWD of candidate documents are therefore necessary to ensure that the retrieved texts remain adequately challenging. We compare two approaches for this task.

5.1 Open learner model (OLM) approach

In the OLM approach, users are expected to manually update the vocabulary set described in Section 3.2. Hence, user update frequency crucially affects retrieval performance. If a user reports all newly acquired word to the OLM before using the text retrieval system, there would be no NWD Error. In practice, the user will likely update these words only after he or she encounters them when reading a recommended document \( d_i \). At the time of search, the newly learned words would remain outside the vocabulary set and contribute to the NWD Error. We experimented with two update frequencies:

**Full Update** This frequency models the conscientious user who updates all words (that require update) when he or she reads the top-ranked documents \( d_i \). Hence, following the \( i \)th search, the vocabulary set is updated as follows:

\[
\text{voc}(\hat{u}_{i+1}) \leftarrow \text{voc}(\hat{u}_i) \cup \left( d_i \cap \text{voc}(u_i) \right)
\]  

**Occasional Update** This models the more casual or conservative user who performs update on only half of the words in \( d_i \) that require update. In this more realistic scenario, newly learned words may remain excluded from the vocabulary set \( \text{voc}(\hat{u}_i) \) even after the user has read them in multiple documents.

5.2 Graded Approach

Akin to a graded reader, the graded approach relies on the user to choose his or her grade, and assigns the vocabulary list corresponding to that grade as the vocabulary set. To create a strong baseline, we assume that the user always chooses the optimal grade. More formally, at time \( i \), the graded approach uses the vocabulary list \( V_\ell \) that achieves the highest F-measure for the ground-truth vocabulary set \( \text{voc}(u_i) \). In our case, then, the graded approach uses \( V_6 \) in the first half of simulation, and then switches to \( V_6 \) at the optimal time, when the user’s lexical knowledge becomes closer to level 6.

This set-up gives the graded approach several advantages over the OLM. The graded approach not only selects the optimal grade, but also “knows” the words that the user will be learning in the simulation (namely, those in \( V_6 \)). The OLM, in contrast, has no access to \( V_6 \) and relies only on user updates. Our simulation result will gauge the amount of user update necessary to reap the benefits of OLM over the graded approach.

6 Results

We conducted the simulation with a database of 1923K Chinese Wikipedia entries and 29K short essays.\(^1\) We performed automatic word segmentation on all documents with the Stanford CoreNLP parser (Manning et al., 2014). We now present a chronological analysis of the simulation (Section 6.1), and then examine the overall experimental results (Section 6.2).

6.1 Chronological analysis

Figure 1 plots the Actual NWD of the top-ranked documents \( d_i \) over the course of the simulation, with the Target NWD set to 20%.

**OLM approach (Full Update).** Throughout the simulation, the model retrieved documents whose Actual NWD was relatively close to the 20% target, with the NWD Gap never exceeding 2%. The

\(^1\)The short essays were downloaded from the website duanmeiwen.com
Actual NWD was consistently below the target because the user would not update the status of newly learned words until he or she reads them in $d_i$.

**Graded approach.** In contrast, the NWD Gap for the graded approach reached a maximum of 5%. The Actual NWD of the recommended documents was initially close to the 20% target. With the vocabulary set kept constant at HSK level 5 during the first half of the simulation, the user’s vocabulary acquisition led to a widening of the NWD Gap, up to 5% towards the middle of the simulation. At this point, with the promotion of the user to level 6, the NWD spiked to as high as 24.0%. The Actual NWD then gradually converged back to the 20% target towards the end of the simulation.

### 6.2 Overall results

Table 1 reports the average NWD Gap and NWD Error over the entire simulation. The OLM outperformed the graded approach at both update frequencies (full and occasional) and at all three targets (20%, 30%, 40%). We first analyze the results at 20% Target NWD, and then examine the effects of higher targets.

**Target NWD at 20%**. The OLM achieved the smallest NWD Gap with Full Update, at only 0.55% below the target. Aided by incremental adjustment to the vocabulary set, it more accurately re-estimated the user’s current vocabulary competence, which in turn led to better NWD estimation.

Occasional Update made the OLM more prone to over-estimate the difficulty of the documents, and hence produced a larger gap (0.84%). The graded approach incurred the largest NWD Gap, with an average of 1.69%. The gap was largest when the user was half-way between levels 5 and 6, since it was forced to choose one of the two and could not offer a middle ground.

In terms of NWD Error, the OLM also outperformed the graded approach at both update frequencies. The OLM was able to retrieve documents whose NWD more closely fits the user’s target.

**Effects of higher Target NWD.** At higher NWDs, a larger pool of candidate documents can

<table>
<thead>
<tr>
<th>Approach</th>
<th>Target</th>
<th>NWD Gap</th>
<th>NWD Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graded</td>
<td>20%</td>
<td>1.69%</td>
<td>1.35%</td>
</tr>
<tr>
<td>OLM (Occasional)</td>
<td></td>
<td>0.84%</td>
<td>0.84%</td>
</tr>
<tr>
<td>OLM (Full)</td>
<td></td>
<td><strong>0.55%</strong></td>
<td><strong>0.55%</strong></td>
</tr>
<tr>
<td>Graded</td>
<td>30%</td>
<td>2.27%</td>
<td>2.24%</td>
</tr>
<tr>
<td>OLM (Occasional)</td>
<td></td>
<td>1.26%</td>
<td>1.26%</td>
</tr>
<tr>
<td>OLM (Full)</td>
<td></td>
<td><strong>0.84%</strong></td>
<td><strong>0.84%</strong></td>
</tr>
<tr>
<td>Graded</td>
<td>40%</td>
<td>2.61%</td>
<td>2.59%</td>
</tr>
<tr>
<td>OLM (Occasional)</td>
<td></td>
<td>1.44%</td>
<td>1.44%</td>
</tr>
<tr>
<td>OLM (Full)</td>
<td></td>
<td><strong>1.07%</strong></td>
<td><strong>1.07%</strong></td>
</tr>
</tbody>
</table>

Table 1: New-word density (NWD) Gap and NWD Error of the top-ranked documents returned by the graded approach and the open learner model (OLM)
fit the search criteria. In general, a document with more difficult words exposes the OLM to more chances of failing to recognize the user has learned those words. As a result, the higher the Target NWD, the larger the NWD Gap, i.e., the farther the recommended documents fell short of the target. For the OLM with Full Update, NWD Gap increased from 0.55% (Target=20%) to 0.84% (Target=30%) and 1.07% (Target=40%). A similar increase can be observed in the Occasional Update setting. These experimental results suggest that text retrieval is more challenging when the user requests documents with more advanced vocabulary.

7 Conclusions

Automatic text retrieval supports language learners in self-directed reading and independent learning. A major challenge in this task is to match learners with different capabilities to texts with appropriate vocabulary complexity.

We have evaluated an open learner model (OLM) that allows users to update their individual progress in vocabulary acquisition. We compared this model to the graded approach, where the system recommends texts to users at the optimal grade. We conducted a simulation of a learner of Chinese as a foreign language who uses the retrieval system during a period of vocabulary acquisition. Results show that the OLM outperforms the graded approach in retrieving texts at a range of target NWDs. When the user makes at least half of the expected updates, the OLM’s fine-grained, incremental adjustment yields superior retrieval performance.

We believe these results can help inform the design of text retrieval systems for language learners. In future work, we intend to further improve retrieval quality by extending the OLM beyond vocabulary to other dimensions of text difficulty, such as syntactic and semantic complexity.

Acknowledgments

We gratefully acknowledge support of the Innovation and Technology Fund (Ref: ITS/389/17) of the Innovation and Technology Commission, the Government of the Hong Kong Special Administrative Region; and of the Hong Kong Institute for Data Science (Project #9360163) at City University of Hong Kong.

References


Transforming Multi-Conditioned Generation from Meaning Representation

Joosung Lee
Kakao Enterprise Corp., South Korea
rung.joo@kakaoenterprise.com

Abstract

Our study focuses on language generation by considering various information representing the meaning of utterances as multiple conditions of generation. Generating an utterance from a Meaning representation (MR) usually passes two steps: sentence planning and surface realization. However, we propose a simple one-stage framework to generate utterances directly from MR. Our model is based on GPT2 and generates utterances with flat conditions on slot and value pairs, which does not need to determine the structure of the sentence. We evaluate several systems in the E2E dataset with 6 automatic metrics. Our system is a simple method, but it demonstrates comparable performance to previous systems in automated metrics. In addition, using only 10% of the dataset without any other techniques, our model achieves comparable performance, and shows the possibility of performing zero-shot generation and expanding to other datasets.

1 Introduction

In many conversation systems, generating sentences with specific information is useful. For example, it can be used in chatbot systems or spoken dialogue systems to generate utterances that contain meaning representations (MRs) corresponding to a user’s query. In order to train the NLG system that reflects this variety of information, a large amount of labeled data is required. At the 2017 E2E challenge (Dušek et al., 2019), a large dataset was released, which consisted of pairs of MRs representing restaurant reviews and corresponding utterances. Table 1 shows an example. MRs can be regarded as the multi-conditions type of utterance generation, which consists of slots and values, and the corresponding utterances are references written by humans. We focus on training the model to generate utterances directly from MRs.

Many of the previous NLG research are a two-stage approach through sentence planning and surface realization. Sentence planning determines the overall sentence structure and surface realization is the process of flattening the sentence structure. In recent studies (Konstas and Lapata, 2013; Dušek and Jurčiček, 2016; Juraska et al., 2018), these two stages are processed at once by learning end-to-end without aligned data with a neural network.

When generating sentences from an input (flat or structured MR), there are a template-based approach and a neural network-based approach. Smiley et al. (2018); Puzikov and Gurevych (2018); Wiseman et al. (2018) generate sentences based on template. The template method is to obtain the structural sets of sentences corresponding to MRs from training data and apply the template appropriate to the test data to generate the sentences. In Smiley et al. (2018); Puzikov and Gurevych (2018), a template is formed based on rules, and Wiseman et al. (2018) learns the template structure of a sentence as a neural network.

There is also a way to generate a natural language with a neural network without using a template. Dušek and Jurčiček (2016); Smiley et al. (2018); Puzikov and Gurevych (2018); Juraska et al. (2018); Elder et al. (2019); Gehrmann et al. (2018) are sequence-to-sequence models of encoders and decoders, which have a one-stage framework, and

<table>
<thead>
<tr>
<th>MR (slot=value)</th>
<th>Utterance</th>
</tr>
</thead>
<tbody>
<tr>
<td>name=Giraffe, eatType=pub, food=Fast food, area=riverside, familyFriendly=yes</td>
<td>On the riverside the Giraffe is a Fast food, kid friendly pub.</td>
</tr>
</tbody>
</table>

Table 1: An example of an E2E dataset consists of pairs of a MR and an utterance.
Dušek and Jurčíček (2016) is the model used by E2E challenge organizers. In Balakrishnan et al. (2019), the encoder and decoder model converts flat MRs to structure MRs, and outputs the sentence through constrained decoding.

Template and neural network methods each have advantages and disadvantages. The template method guarantees a certain performance but limits the diversity and possibilities of the output. The neural network method generally performs better than the template method, but requires a lot of data and has limitations in the naturalness and semantical correctness of sentences (Nayak et al., 2017).

We propose a novel approach using the Transformer decoder as a simple one-stage framework. In our model, GPT2-small (Radford et al., 2019) is the backbone, and \((s_i, v_i)\) pairs of the meaning representation are put into multiple conditions to generate a sentence. In the previous works (Juraska et al., 2018; Balakrishnan et al., 2019; Smiley et al., 2018; Puzikov and Gurevych, 2018; Dušek and Jurčíček, 2016), when receiving the meaning representation as input, the value is delexicalized. Specifically, all values corresponding to the same slot are delexicalized to a placeholder so that unseen inputs can be processed. However, Nayak et al. (2017); Juraska et al. (2018) report that delexicalization often leads to inappropriate behavior in scenarios. For example, the word "cheap" can be reflected in the utterance that matches the value of "less than $20". Also, when food[Italian] is given as slot[value], "Italian food" is an appropriate phrase in a generated utterance, but in the case of food[fast food], "fast food food" is an incorrect phrase. Additionally, the combination of eatType[coffee shop] and food[Italian] is rather weird, and the combination of name[The Rice Boat] and area[riverside] is appropriate, so delexicalization doesn’t fully utilize the characteristics of tokens. We, therefore, treat the slot as a special token and the value as a regular token of vocabulary without delexicalization in the training. Nevertheless, in testing for unseen values, the model generates appropriate utterances and is described in Section 4.3. Our method is considered as generating a sentence as a simple one-stage framework directly from flat MRs.

Our model is tested on the E2E dataset. In addition to the evaluation metrics used in the E2E challenge, the systems are evaluated with BERTScore (Zhang* et al., 2020). Our approach shows the best performance in BLEU, METEOR, and BERTScore and competitive performance in other metrics. By leveraging the pre-trained model GPT2, we quickly converge the model with only a few epochs and generate fluent utterances without considering the structure of the sentence. In addition, even if only 10% of the training data is used, it achieves performance comparable to previous systems.

2 Related Work

In many NLP tasks, the Transformer-based (Vaswani et al., 2017) models have recently shown good performance. Gehrmann et al. (2018) is a previous work using the Transformer encoder and decoder in the E2E task. BERT (Devlin et al., 2019) composed only of the Transformer encoder is used a lot in NLU tasks, and GPT (Radford et al., 2019) composed only of the Transformer decoder is used a lot in NLG tasks. These models are trained as large-scale open-domain corpora. By leveraging pre-trained models trained on large datasets and applying them to downstream tasks, many NLP tasks achieve better performance.

The generation of utterances from MRs is quite similar to machine translation, one of the sequence-to-sequence tasks. Also, in terms of generating sentences with certain restrictions, it is similar to style transfer (Logeswaran et al., 2018; Lample et al., 2019; Lee, 2020), which is one of sequence+condition-to-sequence tasks. However, the generation of utterances from MRs is not exactly the same as the above tasks because of the condition-to-sequence perspective. Since the E2E task does not have a given sequence as the input of the model, we approach the sentence generation using only the decoder without sequence encoding. We choose a language model of the Transformer decoder that performs better than LSTM and uses the pre-trained model GPT2 as a backbone.

3 Our Approach

3.1 Problem Statement

E2E dataset The domain of the E2E dataset is a restaurant and consists of \(D = \{(M_1, u_1), \cdots, (M_n, u_n)\}\). \(M_i\) is the MR and contains the (slot, value) pair, \((s_i, v_i)\), and \(u_i\) is the corresponding utterance. There are \(8\) types of slots, and the value corresponding to each slot has various numbers (2 to 34). Statistics of the overall dataset are shown in Table 2.
### Table 2: Statistics of E2E dataset provided by Dušek et al. (2020).

<table>
<thead>
<tr>
<th></th>
<th>E2E Dataset</th>
<th>MRs</th>
<th>References</th>
<th>Slots/MR</th>
<th>Tokens/Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>training</td>
<td>4,862</td>
<td>42,061</td>
<td>5.52</td>
<td>20.27</td>
<td></td>
</tr>
<tr>
<td>development</td>
<td>547</td>
<td>4,672</td>
<td>6.3</td>
<td>24.52</td>
<td></td>
</tr>
<tr>
<td>test</td>
<td>630</td>
<td>4,693</td>
<td>6.91</td>
<td>26.76</td>
<td></td>
</tr>
</tbody>
</table>

The goal of the task is to generate a $u_i$ by reflecting $M_i$. $s_i$ is the concept of a given category of conditions, and $v_i$ is an item that should actually be reflected in utterance generation. Since utterances can also reflect values as synonyms, changing them to a single placeholder is considered a risk that the processing of input and output will not work properly. In model training, not only delexicalization but also preprocessing of special data such as Smiley et al. (2018) is not performed, and the model is expected to learn about tokens and phrases with different inputs and outputs such as synonyms.

### 3.2 Pre-trained Model: GPT2-small

Our system uses GPT2-small model as a backbone to generate utterances and is illustrated in Figure 1. GPT2 is an unsupervised pre-trained model with large-scale open-domain corpora of unlabeled text. GPT2 uses only the Transformer decoder and generates sentences from left to right autoregressively without an encoder. GPT2-small has 768-dimensional embedding size, 12 heads, and 12 layers, so the total number of parameters is 117M. Our system has the advantage of starting from knowing the token distribution by using the pre-trained model as the initial state.

### 3.3 Generation

As conditions of an utterance generation, $(s_i, v_i)$ pairs of a flat MR are given, slots are treated as special tokens. Specifically, the tokens for slots and start are added to vocabulary, and the regular tokens in vocabulary are used for values and end. In order to distinguish between special tokens and regular tokens, special tokens are changed to $<$SPECIAL TOKEN$>$ to form vocabulary. Inputs are given by concatenating the given $(s_i, v_i)$ pairs in order (e.g. $<$name$>$, Giraffe, $<$eatType$>$, pub, ...). We use $<$name$>$, $<$eatType$>$, $<$food$>$, $<$priceRange$>$, $<$customer rating$>$, $<$area$>$, $<$familyFriendly$>$, $<$near$>$ in the fixed order of slots as formed in the E2E dataset, and do not put slots not in the given MR as input. Our system generates utterances by considering the MR as multi-conditions of generation. Through training end-to-end Multi-Conditioned generation, we hope that the model find out the role of a MR.

Transformer Decoder is autoregressively unidirectional from left to right, so the model only sees the previous tokens:

$$\alpha_i = \text{TransformerDecoder}(u_1^{i:k}, \langle \text{start} \rangle, s_i, v_i)$$

where $u_1^{i:k}$ denotes tokens up to $k$th tokens of $u_i$, $\alpha_i$ is the output of the Transformer Decoder, $k \times h$ tensor, and $h$ is the hidden embedding dimension of the decoder.

To predict the token of the next step $(k + 1)$, multiply the $k$th-vector $\alpha_i^k$ by matrix $M$ as follows:

$$u_i^{k+1} = \text{argmax}(M(\alpha_i^k))$$

where $M$ is a randomly initialized matrix.

Because our system is a one-stage framework that generates utterance directly from flat MRs, sentence planning and surface realization are not considered separately. Our approach shows strong results in 4.2 without additional techniques such as delexicalization, data augmentation, and extra datasets. When testing, it is possible to deal with unseen values by delexicalization, which is described in detail in Section 4.3.

### 3.4 Training

We experiment using one V100 16GB GPU in Linux environment on an AWS server. Our system is end-to-end trained with the AdamW (Loshchilov and Hutter, 2019) optimizer for 5 epochs. The initial value of the learning rate is 2e-5 and is adjusted with a linear scheduler. The model is trained so that the output at the current step $(k)$ predicts the token of the next step $(k + 1)$ and the loss of the objective function is calculated as:

$$L(\theta) = - \sum_{(M_i, v_i) \in D} \log p(u^1_i | s_i, v_i) + \sum_{k=1} \log p(u_i^{k+1} | u_i^{1:k}, s_i, v_i) + \log p(\langle \text{end} \rangle | u_i^{1:k}, s_i, v_i)$$

where $u_i^k$ is the $k$th tokens of $u_i$ and $\langle \text{S}\rangle$, $\langle \text{E}\rangle$ are start, end token respectively. Since the ground truth given in the dataset is only utterance, the outputs before $\langle \text{S}\rangle$ is entered cannot be used for loss calculation.

During training time, tokens are generated by applying teacher-forcing, and tokens are generated by self-feeding during testing time.
Figure 1: Our structure has GPT2 backbone based on the Transformer decoder. Blue tokens are special tokens that are newly added to the vocabulary. The model autoregressively starts to generate utterance from when the \texttt{<START>} token is received as input, and ends when the \texttt{<endoftext>} token is output.

4 Experiments

We use the model provided by HuggingFace \footnote{https://huggingface.co/gpt2} to make it easy to use the pre-trained GPT2 trained by OpenAI.

4.1 Evaluation Metrics

We use the five automatic evaluation metrics used in the E2E Challenge, BLEU (Papineni et al., 2002), NIST (Lin and Och, 2004), METEOR (Denkowski and Lavie, 2014), ROUGE (Lin, 2004) and CIDEr (Vedantam et al., 2015), equally as the basis. Evaluation scripts are provided by challenge organizers \footnote{https://github.com/tuetschek/e2e-metrics}. We additionally calculate the similarity F1 scores of the two sentences using the BERTscore of RoBERTa model (Liu et al., 2020) provided by the library \footnote{https://pypi.org/project/bert-score/}. The two sentences entered in the BERTscore library are the generated utterances and human references. The BERTScore metric is task agnostic and, unlike previous metrics, uses importance weighting between contextual embedding. Therefore, it is a common metric that calculates a better correlation by solving the disadvantages of the previous metric. BERTScore is measured only for the systems that provided the output for the test dataset.

4.2 Results

Table 3 shows the experimental results of our system and comparison systems. The first section is our model \textit{Multi-Conditioned Transformer}, which consists of the Transformer decoder.

4.2.1 Compared Systems

TGen (Dušek and Jurčiček, 2016) is the baseline tested by the E2E challenge organizer. SeqGen (Smiley et al., 2018) is the system that participated in the challenge, and Slug2Slug (Juraska et al., 2018) is the system that won the E2E challenge. Slug2Slug improves performance by learning the surface realization model as additional data and ensemble the three models. Model-T (Puzikov and Gurevych, 2018) and TempleGen (Smiley et al., 2018) are rule-based systems using templates. NTemp + AR (autoregressive) (Wiseman et al., 2018) is a hidden semi-markov model (HSMM) decoder that learns the structure of a template. Template-based systems guarantee a certain quality and fluency of natural language generation, but overall performance is lower than neural networks. Dot-copy and Transformer (Gehrmann et al., 2018) are methods of learning the structure of a template with a neural encoder and decoder. The hyperparameter $K$ of these two systems indicates the number of models to be diverse ensembling. TripAdvisor (Elder et al., 2019) follows a two-stage approach: (1) content selection at the system input to generate a symbol intermediate representation and (2) generating utterance. Each stage proceeds with the structure of a neural encoder and decoder and improves the performance of the model with additional data.

4.2.2 Automatic Evaluation

The performance of our system and the comparison systems are shown in Table 3. In these systems, Multi-Conditioned Transformer achieves the best performance in BLEU, METEOR, and BERTScore, and the second-best in NIST. Our system also shows competitive performance compared to previous systems in ROUGE and CIDEr metrics. We experimented with at least 3 random seeds and observe that our model always reaches similar performance.

"No pre-trained" is a model trained from scratch and the rest are the same except for initialization. If our model is trained without the pre-trained tech-
### Table 3: Automatic metric scores of our and compared systems in the E2E test dataset. Systems are evaluated with BERTscore along with the five metrics used in the E2E challenge. The number in parentheses is the standard deviation. The bold number is a notation for the best performing system.

<table>
<thead>
<tr>
<th>System</th>
<th>BLEU</th>
<th>NIST</th>
<th>METEOR</th>
<th>ROUGE-L</th>
<th>CIDEr</th>
<th>BERTscore</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-Conditioned Transformer</td>
<td>0.6794</td>
<td>8.6477</td>
<td>0.4579</td>
<td>0.6998</td>
<td>2.2884</td>
<td>0.942</td>
</tr>
<tr>
<td>30% sampling (avg)</td>
<td>0.6651</td>
<td>8.5712</td>
<td>0.4364</td>
<td>0.6871</td>
<td>2.1561</td>
<td>0.940</td>
</tr>
<tr>
<td>10% sampling (avg)</td>
<td>0.6541</td>
<td>8.4332</td>
<td>0.4271</td>
<td>0.6761</td>
<td>2.0786</td>
<td>0.939</td>
</tr>
<tr>
<td>No pre-trained</td>
<td>0.5885</td>
<td>8.0320</td>
<td>0.3962</td>
<td>0.6302</td>
<td>1.7585</td>
<td>0.930</td>
</tr>
<tr>
<td>Tgen (baseline)</td>
<td>0.6593</td>
<td>8.6094</td>
<td>0.4483</td>
<td>0.685</td>
<td>2.2338</td>
<td>0.939</td>
</tr>
<tr>
<td>Model-T</td>
<td>0.6567</td>
<td>7.4544</td>
<td>0.4529</td>
<td>0.6614</td>
<td>1.8206</td>
<td>0.938</td>
</tr>
<tr>
<td>Slug2Slug</td>
<td>0.6619</td>
<td>8.613</td>
<td>0.4454</td>
<td>0.6772</td>
<td>2.2615</td>
<td>0.942</td>
</tr>
<tr>
<td>TemplGen</td>
<td>0.4202</td>
<td>6.7686</td>
<td>0.3968</td>
<td>0.5481</td>
<td>1.4389</td>
<td>-</td>
</tr>
<tr>
<td>SeqGen</td>
<td>0.6336</td>
<td>8.1848</td>
<td>0.4322</td>
<td>0.6828</td>
<td>2.1425</td>
<td>-</td>
</tr>
<tr>
<td>NTemp+AR</td>
<td>0.598</td>
<td>7.56</td>
<td>0.3875</td>
<td>0.6501</td>
<td>1.95</td>
<td>-</td>
</tr>
<tr>
<td>dot, copy, K = 2</td>
<td>0.674</td>
<td>8.61</td>
<td>0.452</td>
<td>0.708</td>
<td>2.31</td>
<td>-</td>
</tr>
<tr>
<td>Transformer, K = 2</td>
<td>0.662</td>
<td>8.6</td>
<td>0.457</td>
<td>0.704</td>
<td>2.34</td>
<td>-</td>
</tr>
<tr>
<td>TripAdvisor</td>
<td>0.6738</td>
<td>8.7277</td>
<td>0.4572</td>
<td>0.7152</td>
<td>2.2995</td>
<td>-</td>
</tr>
</tbody>
</table>

### Table 4: Comparison of utterances for given MRs. Samples given according to when 3 and 7 \( (s_i, v_i) \) pairs are given. In the E2E dataset, the human reference provides several versions but extracts one sample.

<table>
<thead>
<tr>
<th>MR</th>
<th>name</th>
<th>eatType</th>
<th>area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-Conditioned Transformer</td>
<td>Blue Spice</td>
<td>pub</td>
<td>riverside</td>
</tr>
<tr>
<td>Tgen (baseline)</td>
<td>Blue Spice is a pub by the riverside.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model-T</td>
<td>Blue Spice is a pub located in the riverside area.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slug2Slug</td>
<td>Blue Spice is a pub in the riverside area.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference sample</td>
<td>There is a pub Blue Spice in the riverside area.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MR</td>
<td>name</td>
<td>eatType</td>
<td>food</td>
</tr>
<tr>
<td>Multi-Conditioned Transformer</td>
<td>The Mill</td>
<td>restaurant</td>
<td>English</td>
</tr>
<tr>
<td>Tgen (baseline)</td>
<td>The Mill is a family-friendly english restaurant in the city centre near Raja Indian Cuisine.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model-T</td>
<td>The Mill is a family-friendly restaurant which serves English food in the price range of less than £20. It is located in the city centre area, near Raja Indian Cuisine.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slug2Slug</td>
<td>The Mill is a family friendly English restaurant in the city centre near Raja Indian Cuisine. It has a price range of less than £20.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference sample</td>
<td>The Mill is a restaurant and is family-friendly, cheap and reasonable priced is very good for the family. We provide full English food. Located near Raja Indian Cuisine In the city centre.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 5: Average of three workers’ ratings

<table>
<thead>
<tr>
<th>model</th>
<th>quality</th>
<th>naturalness</th>
</tr>
</thead>
<tbody>
<tr>
<td>ours</td>
<td>4.525</td>
<td>4.625</td>
</tr>
<tr>
<td>TGen</td>
<td>4.317</td>
<td>4.498</td>
</tr>
<tr>
<td>Slug2Slug</td>
<td>4.340</td>
<td>4.545</td>
</tr>
</tbody>
</table>

4.2.3 Human Evaluation

Human evaluation is performed for quality and naturalness as in Dušek et al. (2020); Juraska et al. (2018), and the results are shown in Table 5. Quality is a score for grammatical correctness and whether generated utterance properly reflects given MRs. Naturalness is a rating of the possibility that utterance is written by a native speaker, regardless of the MRs. We randomly sampled 200 samples from our test set and hired 3 workers from Amazon Mechanical Turk 4 to rate them on a scale of 1(bad)-5(good). Slug2Slug is a system that ranked first and second in quality and naturalness, respectively, in the E2E challenge. In human evaluation, our model shows better than baseline TGen and Slug2Slug.

---

4https://www.mturk.com/
4.2.4 Utterance Generation

Table 4 shows the output of the systems. As with BERTscore calculations, other comparative systems with no output provided cannot verify the utterances. When there are three \((s_i, v_i)\) pairs, there is no difference between our system and previous systems. However, as the number of pairs increases, the possible sentence structures vary, so different systems output different utterances. Test data with many pairs is considered to make a difference in automatic evaluation performance. Since our system is based on the Transformer, it can be more robust and general to long-term sequences than LSTM-based systems.

4.2.5 Training with Less Data

The 2nd and 3rd rows of Table 3 are the results of fine-tuning our model by sampling only a small amount of training dataset. For small training data, 10% and 30% of the entire training dataset are randomly sampled. Our model is trained and averaged by performing random sampling three times in consideration of the possibility that the performance of the model may vary according to the statistics of the randomly sampled dataset. We found that the performance of our model was similar even with random sampling. Our system shows a similar level of performance with a small standard deviation according to the sampled data.

As the sampling of the training data of the system increases, the performance of the system improves. However, if we use more than 50% of the training data through many experiments, our system has little improvement in performance. Our approach can leverage the pre-trained language model to take advantage of the background knowledge of sentence generation. Therefore, it is difficult to expect a linear relationship between increasing the number of training data and increasing performance. However, with the effect of background knowledge, our system, which was trained by sampling 10% of the training data, shows performance comparable to previous systems. The performance of the model trained by sampling 30% of the data is similar to that of the Slug2Slug, the system that won the E2E challenge. Building our system with only 30% of the training data and showing good results demonstrates the effectiveness of using the pre-trained model. If a better pre-trained model is used as the backbone, we hope to build an effective model with less data.

4.3 Generation from Unseen Values

Our model is not trained on unseen values, so it can have weaknesses in real applications. Therefore, we introduce a zero-shot generation method through Sim-Delexicalization. Table 6 shows an example of this experiment. The value of \(<\text{familyFriendly}>\) is not treated as unseen value because it only has \((\text{yes or no})\). Our system generates proper utterance through the following two steps for zero-shot generation.

1. **Sim-Delexicalization:** The given unseen values are replaced with similar values among the lists of value corresponding to the same slot. In the first example of Table 6, "Green Man" is replaced by "Blue Man" and "2.1 out of 5" is replaced by "3 out of 5". In the second example we observe that expensive is replaced by high. Also, taking into account the grammatical aspect, if an unseen value containing "the" in the \(<\text{name}>\) and \(<\text{near}>\) slots are given, the value list containing "the" is limited as a candidate (and vice versa). There can be several ways to find similar tokens, but we use BERTscore to select the value with the highest score.

2. **Relexicalization:** Replaced values are changed back to unseen values in generated utterances. "Green Man" is deciphered as "Blue Man" and other values proceed as well.

The generated utterances are of appropriate quality from a human perspective. In the previous study, unlike delexicalization of unseen values to one placeholder, we have the difference of converting to similar values. Changing to one placeholder in the test also has the same risk as in training above, so we used the existing list of values to change it to an appropriate value each time. In other words, our system can generate utterances that are suitably customized for a given \((s_i, v_i)\). Rather than using only BERTscore, it may be helpful to find similar words using word embedding techniques such as Glove (Pennington et al., 2014) and FastText (Bojanowski et al., 2017) but this will be left for further study.

4.4 Experiments on a Different Dataset

Our paper focuses on the E2E dataset, but for the possibility of scaling, we do a simple experiment in the WebNLG challenge task (Colin et al., 2016) similar to the E2E dataset with the same approach. The WebNLG dataset is collected from DBpedia, and the train, development, and test datasets are
Table 6: Zero-shot generation from unseen values. Given unseen values, the system generates an utterance subject to delexicalization of unseen values to similar seen values.

Table 7: Example of the WebNLG dataset. In one sample, the category is fixed as Airport, and multiple values corresponding to (subject, property, object) can be given.

Table 8: Comparison of systems evaluated with the WebNLG dataset. It was evaluated using the same library as the E2E dataset.

6940, 872, and 1862, respectively, and examples are shown in Table 7. It is significantly smaller than the E2E dataset, and MRs consisting of values corresponding to four (category, subject, property, object) slots are given as a condition. In other words, unlike E2E, the WebNLG dataset has 4 fixed slots, but multiple values can be given.

Table 8 shows the experimental results, and the three comparison systems that participated in the challenge (Gardent et al., 2017) are as follows: (1) The baseline is a neural system trained with OpenNMT. (2) Melbourne shows the best score for all automatic evaluations in the WebNLG dataset, and MRs consisting of values corresponding to four (category, subject, property, object) slots are given as a condition. In other words, unlike E2E, the WebNLG dataset has 4 fixed slots, but multiple values can be given.

Table 8 shows the experimental results, and the three comparison systems that participated in the challenge (Gardent et al., 2017) are as follows: (1) The baseline is a neural system trained with OpenNMT. (2) Melbourne shows the best score for all automatic evaluations in the challenge with an end-to-end LSTM with attention model. Performance is improved by preprocessing entity tagging by collecting information from DBPedia. (3) UPF-FORGe (Mille et al., 2017) is a grammar-based NLG system and has the highest score in human evaluation through rule-based graph-transducers for syntactization.

The systems are evaluated in the same way as the E2E dataset and our system is better than the baseline but slightly worse than the best systems in many metrics. However, our system shows meaningful results in BERTscore and is a simple method that utilizes a pre-trained model without any rule definition, preprocessing, or other datasets. If we preprocess the data like the comparison systems above, our model can expect better performance.

4.5 Analysis
Section 4.3 experimentally demonstrates that our approach is capable of zero-shot generation, a situation not found in the training. The system is trained to generate appropriate utterances for \((s_i, v_i)\) without the delexicalization. Therefore, there is no need to take the ambiguous risk of replacing unseen values with a single delexicalization of placeholders. Instead we introduce sim-delexicalization, which allows the system to reflect unseen values.

The two-stage framework needs to reveal the structure of the sentence, so it is difficult to solve as the number of \((s_i, v_i)\) pairs increases. However, our approach is easily extensible for more pairs. In the WebNLG dataset, we show that it is possible to extend multiple values as well. Since only slots are replaced with special tokens and values are used as regular tokens, our system can be trained to learn the utterance corresponding to MRs without limiting the number of pairs.

We also conducted experiments using a larger backbone model, GPT2-large, but the change in performance is small.

5 Conclusion
This paper presents a simple one-stage approach to generating natural utterances from flat MRs. Our system uses a pre-trained model language model to improve the performance of the system and shows that it is better than the system that won the challenge in human evaluation. Even if our model is trained with only a small amount of sampling data due to the leveraging effect, it is comparable to the previous models. Our approach is simple, efficient, easy to extend to multiple MRs (i.e. WebNLG), and enables zero-shot generation without additional data through sim-delexicalization.
References


Frustration Level Annotation in Latvian Tweets with Non-Lexical Means of Expression

Viktorija Leonova¹, Janis Zuters²
Department of Computer Science, University of Latvia, 19 Raina Blvd., Riga, Latvia
¹viktorija.leonova@gmail.com
²janis.zuters@lu.lv

Abstract

We present a neural-network-driven model for annotating frustration intensity in customer support tweets, based on representing tweet texts using a bag-of-words encoding after processing with subword segmentation together with non-lexical features. The model was evaluated on tweets in English and Latvian languages, focusing on aspects beyond the pure bag-of-words representations used in previous research. The experimental results show that the model can be successfully applied for texts in a non-English language, and that adding non-lexical features to tweet representations significantly improves performance, while subword segmentation has a moderate but positive effect on model accuracy. Our code and training data are publicly available¹.

1 Introduction

Dramatically increasing data storage and processing capacities have resulted in an explosion of available data, and of potential uses for the many kinds of knowledge or insights that could theoretically be extracted from that data. The development of Web 2.0 has created unprecedented amounts of text, and, in recent decades, images and videos. But the sheer volume of the available data is problematic due to a shortage of human resources (time and attention) available for analyzing or even just browsing through it all, as described in (Verma et al, 2016). From the very start, as soon as machine learning techniques appeared, they were immediately applied to text analysis. Neural networks have proven particularly suitable for such tasks, but to learn to approximate human judgements, these methods generally need a large amount of manually annotated training data to obtain a viable model. Creating such corpora is a very time-consuming task, especially if it has to be done from scratch. For languages with a relatively low number of speakers, such as Latvian, this is a particularly pressing problem, because of the small number of textual corpora available, so that researchers generally need to create their own datasets.

Much of the information of importance to be captured from the Internet is by its nature related to emotions. Recommendation systems, conversational “chat bots”, automated customer support assistants, various tools for analyzing, monitoring or enhancing personal well-being or mental health, not to mention systems for targeting or crafting advertising or marketing messages — all can benefit from understanding the emotional state of the recipient or sender of a given message. But recognition of emotions in text generally requires considerable investment of human resources; and is difficult to automate in principle. Perception of emotions is largely subjective, and the emotions perceived from the same source by different people differ. Further, in written communication, and especially communication in social networks, the words alone do not capture all of the emotional information available in the message. For example, the emotional tone of a sentence may vary depending on whether it was followed by period as the absence of periods is a characteristic feature of internet jargon usage, as shown in (Khalifa, 2020), which may carry a different emotional charge.

In this work, we propose the method for automatic annotation of the emotional charge found in text with the help of non-lexical means of expression — typographical marks, emojis and

¹ Source code is available at
https://github.com/Lynx1981/dfrustration/
likewise. In addition, to be able to do so for Latvian, we propose an annotated dataset that was used for model training and its results.

2 Background and Related Works

As machine learning methods, and neural networks, in particular, have developed over the past couple of decades, they naturally started to be applied to recognizing emotions, especially following the demonstrated successes of neural networks in image recognition as in (Giacinto et al., 2016). Emotion recognition in speech, as the easier task, began to appear as early as 2000 in (Nicholson et al., 2020). We begin to see published research on the recognition of emotions in text only beginning with (Alm et al., 2005). The majority of works focus on the classification of text according to its predominant emotion, which means that the text or part of the text was classified as containing one of the basic emotions. The most commonly used is Ekman’s emotion classification scheme, described in (Ekman, 1992), which contains six basic emotions: anger, fear, sadness, disgust, surprise, and joy — either in the standard way, or with various extensions, such as in (Yao et al., 2014), or reductions, as in (Lee and Wang, 2015).

Another model used to classify emotions, in two variants, is the Russell circumplex model, described in (Russell and Mehrabian, 1977), which is a two- or three-factor model where each emotion is represented in a two-dimensional (valence-arousal) or three-dimensional space, with axes of dominance, valence and arousal. Although there are numerous works that use these emotion classification models — for example, the three-factor model used in (Parthasaraty and Busso, 2017) and the two-factor model in (Yu et al., 2016) — these are not as common as works using a categorical list of basic emotions.

As computing power and data grew, the classification of emotions began to shift from a purely qualitative to a more quantitative approach: the intensity of emotions, not just the presence of specific emotions in the text, began to be studied. Some authors even create automatic tools for classifying emotions with intensity, for example the Weka package presented in (Bravo-Maquez et al., 2019) for four emotions (anger, fear, joy and sadness), or another opensource emotion computing framework presented in (Duppada and Hiray, 2017) for the same four emotions.

Interestingly, however, none of these classifications include frustration, or dissatisfaction, in the list of emotions, even if the emotion itself is well known to everyone and plays an important role, for example, in assessing quality of service in (Stauss et al., 2005). In customer service, however, ‘dissatisfaction’ is typically used instead of the word ‘frustration’, but this does not change the content of the term. There are very few works on annotating frustration, and they are relatively old, for example, (Klein et al., 2002) and (Hone, 2006), where the authors discussed the possibility of reducing human frustration in dialogue with the help of an emotional or empathic agent. In (Kapoor et al., 2007), the authors achieved good results in identifying frustration, but they used a complex multimodal system that measured the physical parameters of human (in their case, student) behavior, such as the pressure on the chair and the speed of the mouse. In short, the recognition of frustration from text has not been adequately studied. A recent paper (Hu et al., 2018) is a relatively rare example in which intensities for eight differing emotional “tones” (anxious, frustrated, impolite, passionate, polite, sad, satisfied, and empathetic) were annotated; however, the goals and methods of this work and ours were significantly different: while we examine the effect of words and non-lexical means of expression on perceived frustration, they look for correlations between user and support worker emotional tones; also, their approach uses a seq2seq (“sequence to sequence”) neural model, while we employ a model based on an architecturally simpler, fully-connected feedforward neural classification network.

The works we have mentioned have one feature in common: the classification of emotions in them is based on the analysis of words, or lexical means of expression. Several studies have been devoted to compiling lexicons, such as (Staiano and Guerini, 2014) and (Strapparava and Valitutti, 2004), but we have not been able to find a lexicon of non-lexical features. (Aman and Szpakowicz, 2007) mentions the use of non-lexical means of expression for the classification of emotions (using Ekman’s scheme) but does not give a list or description of these means. With the growing popularity of Twitter as a source of data, several works are making use of emojis, like (Wood and Ruder, 2016) or hashtags, like (Das and Bandyopadhyay, 2010). The article (Mohammad...
and Kiritchenko, 2015) is devoted to the construction of a hashtag lexicon for the automatic classification of emotions, however, there is one problem with their use: most Twitter messages do not contain any hashtags, at least in the domain of customer support conversations. Textual features such as the use of exclamation and question marks were used in (Hasan et al., 2014), and message length in (Roberts et al., 2012). The use of non-lexical (and non-linguistic) means of expression for the classification and intensity of emotions is much more developed for voice communication. An example is (Hautasaari, 2019), where both non-lexical features such as speaking speed or number and length of pauses, and non-linguistic features such as inhalation and exhalation are used to classify emotions into eight classes.

Attempts to use other linguistic features, such as the ratio of sentence parts to each other, like in (Devillers and Vidrascu, 2006) or the number of word separators or word separator sequences as in (Perikos and Hatzilygeroudis, 2016) in addition to a basic bag-of-words representation, are being made, but a systematic review and study of such features is still lacking.

3 Preceding Work

We use two gauge points for appraising the performance of our model: first is the baseline accuracy, obtained by always predicting the most frequent ground-truth rating, and second — the results obtained by an equivalent model based solely on lexical features and not employing any sort of input processing. A paper by (Zuters and Leonova, 2020) presented such a model for predicting frustration intensity level based on lexical features only. There, the authors employed a fully connected feedforward neural network with 64 hidden units. This network took as input a bag-of-words representation of the input text, using a subset vocabulary constructed during the training phase. In order to construct this, for every word in the dataset that was encountered in more than 2 entries, the following statistics were calculated: the average value of frustration intensity of the entries this word was found in, and the standard deviation of this value. Entries that were annotated as not rated (“n”) or missing a rating value were ignored. The standard deviation is a main criterion for constructing the bag-of-words “best words” vocabulary, based on reasoning that the lower the standard deviation of the frustration rating, the more characteristic the specific word is for the given frustration intensity.

<table>
<thead>
<tr>
<th>Entry</th>
<th>No. of occur.</th>
<th>Avg. value</th>
<th>St. dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>offer</td>
<td>7</td>
<td>2.5714</td>
<td>0.7284</td>
</tr>
<tr>
<td>offered</td>
<td>3</td>
<td>3.3333</td>
<td>0.4714</td>
</tr>
<tr>
<td>offering</td>
<td>3</td>
<td>3.3333</td>
<td>0.9428</td>
</tr>
</tbody>
</table>

Table 1: Statistical metrics of different forms of the word “offer”.

Table 1 provides an excerpt from such a vocabulary: for each word, the following numbers are provided: the number of occurrences of the word in the dataset, the average value and the standard deviation of this value.

For each set of training data, a vocabulary was constructed, and then used for preprocessing each input entry for calculating predictions. The output of the model was a value from 0 to 4, representing the predicted frustration intensity. The performance of this model is used as a baseline for comparison.

4 Model

Here, we propose a new model that uses as input a set of features based on non-lexical means of expression in addition to the basic bag-of-words representation, and also employs subword segmentation for input preprocessing. By adopting

![Figure 1: Model schema.](image-url)
these techniques, we demonstrate a significant improvement in prediction accuracy, as discussed in detail in the Results section. A schematic of the model is given in Figure 1.

It can be seen that a user message in the model is used to construct two types of features: lexicon-based and non-lexical means of expression-based. We use the same method of lexicon construction as developed for the baseline model which was described in the Previous Work section. However, the new model differs in that it applies subword segmentation of the user message prior to constructing the bag-of-words representation (which thus becomes a ‘bag-of-subword-units’), and it also adds features to the model input based on a range of non-lexical means of expression. We next discuss these in detail.

4.1 Non-Lexical Means of Expression

While there is no shortage of research based on annotating emotions on the basis of lexical means of expression, and considerable effort has been dedicated to developing lexicons and word embeddings to improve the results, non-lexical means of expressions are used but very sparingly. It is true that in text-based social media — as opposed to, for example, in personal conversations — non-verbal signs of emotion such as intonation or facial expressions are naturally absent, and thus mostly lexical means are used for emotion identification from text. However, people creatively use the means available to compensate, at least partially, for the absence of such non-verbal means. To this end, built-in and homemade emoticons, or "smileys", sometimes composed of typographic marks, as well as typographic marks themselves (e.g. quotes for sarcasm), and also hashtags and the like, are very commonly used.

(Mohammad and Bravo-Marques, 2017) showed that hashtags consistently increase the perceived intensity of emotions in Twitter messages in English, which suggests that making use of these and other non-lexical means of expression to improve automatic annotation has promise. We also would like to mention that not all emojis are used equally by all users. Some appear situationally and play an illustrative role by commenting the text in the form of an image. However, emoticons are not the only means of expression used to express emotions in a text. Traditional forms are also used, such as punctuation marks of all kinds, and conversational features such as two-, three- or more -fold repetition of letters, as well as more Internet-specific ones such as uppercase writing, among others.

4.2 Feature Selection

In the dataset we have constructed, we have identified a number of non-lexical means of expression (NLME), for each of which we calculated the correlation with the median human-annotated level of frustration. The correlation served as a selection criterion for selecting NLME for further feature construction. The original correlation table can be found in the accompanying GitHub repository, along with the other source files. Contrary to our expectations, we found that means of expressions such as hashtags do not possess predictive value for the level of frustration.

The same is true for emoticons expressing seemingly positive feelings, such as smiling, laughing faces and similar. Having looked more closely at the examples containing such emoticons, we concluded that the most likely reason for the absence of such correlation is that these are used to denote sarcasm as often as not. An interesting fact is that, while the tendency is preserved for self-made emoticons constructed from typographic marks, such as “(-:”, it is less pronounced. The final list of selected features looks as follows:

- Message length
- Number of exclamation marks
- Number of exclamation marks normalized in relation to the message length
- Number of question marks
- Number of dots
- Number of commas
- Number of quotation marks (single, double, and reversed quotes)
- Number of uppercase words of length 5 and more
- Number of repeated letter “a” sequences
- Number of Twitter built-in emojis
After having selected the promising features, we tested different combinations of those and found the following. Firstly, there is no single feature dominantly responsible for the improved performance. The best feature, namely, the number of exclamation marks, gave only 46.8% accuracy. Secondly, exclusion of the worst features, whose inclusion individually gives worse results than the model without any NLME features at all (quotes, repeating letters, negative smileys made of typographic marks, presence of a picture in the message), also decreases performance by about 0.5%. This means that all the listed features are necessary to achieve the maximal performance.

### 4.3 Segmentation

Another technique that we employed in order to improve performance is preprocessing of the input data with a subword segmentation tool. The reasoning behind this is that Latvian is closer to being a synthetic language than an analytical one, and thus each word is present in the dataset in a multitude of forms — differing for every combination of case, number, gender, and other grammatical categories.

To alleviate this effect, segmentation has been successfully employed in machine translation field. Table 2 shows the top ten entries from the word dictionary, illustrating the principle.

As the vocabulary for subsequent frustration level annotation is constructed automatically based on the distribution of the ratings for specific words, segmentation should facilitate classifying the same words in different grammatical forms together, potentially improving prediction accuracy. It also allows to unify a number of forms under a single entry, reducing data sparsity. As we analyzed these entries in comparison with the whole-word dictionary, we saw, that: 1) brand names (“lg”, “mac”) and unchanged (“nov.”, “neierobežots”) words preserved their place on top; one entry (“-isku”) is a word ending that couldn’t have been in an unsegmented dictionary; two (“publisk-”, “pagāj-”) are new developments — they originally appeared as whole words less than three times and were thus previously ignored, but are now included by virtue of serving as a root form for multiple related word-forms; and, finally, the remaining three (“izmēģin-”, “piezvan-” and “neiet”) were reranked, for similar reasons. The improvements in performance due to subword segmentation are discussed in the Results section.

### 4.4 Additional Processing

In addition to subword segmentation, we also explored other input preparation methods. Specifically, removing diacritical marks in original entries in order to unify spelling variations differing only in their presence or absence, and replacing abbreviations for time, speed and other units of measure, as well as popular sources of spelling variations, with their full forms. The effect of these two methods, even applied cumulatively, was found to be disputable at best and provided, no real improvement of the model accuracy.

### 5 Dataset

In this work, we have used a completely new Latvian dataset, developed specifically for the purpose of testing the performance of our proposed frustration annotation model against the old one used as a gauge point. Following the example of many other recent researchers, we selected Twitter

<table>
<thead>
<tr>
<th>Entry</th>
<th>No. of occur.</th>
<th>Avg. value</th>
<th>St. dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>lg</td>
<td>8</td>
<td>2.0</td>
<td>0.0</td>
</tr>
<tr>
<td>publisk</td>
<td>5</td>
<td>3.0</td>
<td>0.0</td>
</tr>
<tr>
<td>neiet</td>
<td>4</td>
<td>2.0</td>
<td>0.0</td>
</tr>
<tr>
<td>mac</td>
<td>4</td>
<td>3.0</td>
<td>0.0</td>
</tr>
<tr>
<td>piezvan</td>
<td>4</td>
<td>3.0</td>
<td>0.0</td>
</tr>
<tr>
<td>neierobežots</td>
<td>5</td>
<td>2.0</td>
<td>0.0</td>
</tr>
<tr>
<td>īsku</td>
<td>4</td>
<td>3.0</td>
<td>0.0</td>
</tr>
<tr>
<td>izmēģin</td>
<td>4</td>
<td>3.0</td>
<td>0.0</td>
</tr>
<tr>
<td>nov.</td>
<td>4</td>
<td>3.0</td>
<td>0.0</td>
</tr>
<tr>
<td>pagāj</td>
<td>3</td>
<td>3.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 2: Ten best segmented entries. The vertical line | indicates a subword unit that gets joined to whatever precedes it.
as a data source of choice. Four major Latvian internet and telecommunication service provider accounts were chosen for collecting conversations of users with customer support representatives. Those accounts are: (@mans_tet), (@mans_LMT), (@Bitelv) and (@tele2Latvija), which belong to companies Tet, LMT, Bite and Tele2, respectively. To provide for not just the possibility of appraising a frustration level at a certain moment of time, but also to allow studying the dynamics of frustration changes from one user turn to the next, the selected conversations contain no less than two user turns, with at least one customer support turn between those. As a turn we consider a sequence of messages that belong to one party in the conversation (not interrupted by any other users’ messages). An essential criterion was that the dialogs should be in Latvian, as obtaining a dataset in Latvian was the primary goal of collecting this dataset of tweets. The conversations were collected manually, in order to ensure that the criteria are met and that eligible conversations were not excluded just because another user replied to a tweet from the conversation, if such intervention did not actually affect the initial dialog. The resulting dataset consists of 283 dialogs with 688 user turns and 531 customer support representative turns. Of those 688 user turns, 9% had a median frustration value of 0, 19% of 1, 31% of 2, 30% of 3 and, finally, 11% of 4. The resulting collection was post-processed and saved in a unified and anonymized format, as described in the Experimental Setup section.

Each user’s turn in the dataset is followed by three values, representing the level of frustration assigned to this turn by three independent annotators. Each value represents a frustration level measured on a scale of 0 to 4, or can be “n” if the annotator judged that a level of frustration could not be determined from the text of the user message, for example, in case of the user simply stating their address. A final option is a missing value, for example if the text was in a language other than Latvian or could not be understood by an annotator for some other reason.

For English, in order for the results to be comparable with the baseline, we have employed the same dataset that was used in (Zuters and Leonova, 2020). The dataset represents a small subset of the Kaggle Customer Support dataset, where approximately 400 consistent dialogues between a support and a user were isolated and annotated for frustration by three independent annotators on the scale of 0..4. In total, this dataset contained 843 user turns, of which 18% had median frustration value of 0, 15% of 1, 28% of 2, 27% of 3 and 11% of 4. Thus, the baseline accuracy of the most frequent value was 31% for Latvian and 28% for English, and the most frequent frustration values were 2 and 3, respectively.

6 Experimental Setup

The model was implemented on Python as a neural network with RELU activation function and a softmax layer. It is using the following metaparameters: number of hidden units, number of epochs. The model, using both non-lexical and lexical means of expression, whether it was using subword segmentation of the input text or not, was trained on the same set of data.

We tested the model on different vocabulary sizes, fixing the number of epochs as 100 and the number of hidden units as 64, and found that the tendency of vocabulary sizes 50 and 500 to underperform has been preserved. We further explored the effect of hyperparameter changes, and also some additional input processing methods. We tried different network sizes (number of hidden units in the dense feedforward layer), including 32, 128, and 256, and concluded that 32 hidden units were not sufficient, while 128 and 256 gave suboptimal results with 0.75%, 0.5% and 1.5% decline in accuracy, respectively.

The model takes as its input a file with a collection of dialogs between users and customer support representatives in an anonymized format. Information such as Twitter user ids and message ids were removed from the dialogs, and any included sensitive information such as e-mail or customer number, is replaced with generic placeholders, following the example of the Kaggle Customer Support dataset. All user messages in the dataset are tagged with either “USER:” or “SUPP:” to denote whether they belong to a client/user or to the company’s customer support representative, respectively. Consecutive messages coming from a single party are joined together, forming a single “turn” — a sequence of messages uninterrupted by another party, so that the model works under the assumption that each two consecutive messages in a dialog belong to the different parties. For all experiments reported here, we use a single value per turn, calculated as a median of three ratings given by the annotators, which allows keeping the
aggregated annotation value as an integer, thus enabling us to perform classification rather than regression. The number of occurrences of each of the median ratings for values 0 to 4 is [59, 131, 210, 205, 83], respectively, with 2 being the most frequent rating with 210 instances.

To determine the optimal selection of features, used in the model, we have used a two-step process. To appraise the performance of the NLME features, for comparison we used the model with bag-of-words input features only. Figure 2 provides its principal schema of operation.

In the first step, we identified all potential features in the dataset and calculated a correlation table with the median annotated value, leaving only those that had at least a weak correlation. In the second step, we have run the model with a different combination of those features. First, we have used none and all the features for a benchmark, and then tested every feature in isolation to verify that no single feature would give a comparable accuracy. Since none did, we next tried to exclude features that in isolation gave worse results than a model using bag-of-words only. However, the removal of underperforming features (those which in isolation give results below the performance of the bag-of-words-only model) resulted in decreasing the overall performance by 0.4% (z-score = 1.32).

For segmentation of Latvian text, we have applied a GenSeg tool, described in (Zuters and Strazds, 2019) to preprocess the dialog file, so that the input now consisted of the messages in an already segmented form, leaving the rest of the process exactly as before — so that the run of the model on segmented versus unsegmented data differed only in the input file. Having fixed the metaparameters at 64 hidden units and vocabulary size 100, we found that subword segmentation improved the resulting model accuracy by 1.25% (z-score = -4.02).

Table 3 summarizes the results for the most prominent configurations of the features. It can be seen that using the combination of the best features increases the accuracy by 1.5% compared to using the single best feature, while adding the underperforming features improves the result by another 0.4%.

Using input preprocessed with the GenSeg segmentation tool gives the highest performance, yielding a total 49% accuracy, which is an 8% improvement over the old model, of which 1.25% can be attributed to the subword segmentation.

Table 3: Prediction results for the NLME model in comparison with different configurations (for Latvian). C1 - NLME model with all features, C2 - NLME model without subpar features, C3 - NLME model with all features and no segmentation, C4 - NLME model with a single best feature, RM – reference model, BM – baseline model.

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>RM</th>
<th>BM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>48.8</td>
<td>48.4</td>
<td>47.5</td>
<td>46.9</td>
<td>42.2</td>
<td>30.5</td>
</tr>
</tbody>
</table>

7 Results

As our goal was to test improvements on Latvian data, and additionally adapt the original model to the specific challenges of the Latvian language, we created a reference point for comparison by training the baseline neural model using our Latvian language dataset. As discussed in the Preceding Work section, we defined a first baseline for prediction as the accuracy achieved by always assigning the most frequent annotation value in the corpus. However, different from the English dataset, the most frequent value in the Latvian dataset is 2 (with a distribution of values [59, 131, 210, 205, 83]), and the corresponding baseline accuracy is 30.5%. The baseline neural model uses 64 hidden units, vocabulary size 100 and a bag-of-words input representation.
Diacritics removal and unification of abbreviations did not have any discernible effect on the model performance. The best results are achieved using 64 hidden units and vocabulary size 100. Summarizing the results of our experiments, we can conclude that using our proposed model for prediction of frustration intensity significantly increases the accuracy of predictions.

8 Conclusions

In this paper, we have proposed a new neural network-based model for frustration intensity prediction for customer messages in the context of conversations with customer support representatives, as well as a new dataset of such conversations in Latvian, used to train and evaluate the model performance. A baseline model used for comparison employs a bag-of-words representation as input, constructed on the basis of vocabulary that is dynamically built during the training phase. The words selected for inclusion in the vocabulary are the ones that have the least standard deviation for annotated frustration intensity values. Our proposed method differs, first and most importantly, by including also features constructed on the basis of non-lexical means of expression, and, secondly, by performing close-to-morphological segmentation as a preprocessing step to make vocabulary construction more coherent. We also examined a few other methods of input processing, namely, removal of diacritics, and unifying popular variations in spelling, but these did not yield any improvements in results, even when used together.

Performance was assessed by performing a leave-one-out cross exhaustive cross-validation, that is, by computing accuracy (as percentage of correct predictions) obtained after training the model on all data except one entry using this one entry for prediction, and repeating this process for all the entries in turn, averaged across five runs. We compare against a similar neural model that only uses lexical features as input. While such a model achieves a 10% improvement over the baseline accuracy obtained by always predicting the median frustration rating of the dataset, our proposed model achieves an improvement in accuracy of 18% over the baseline and 8% over the old model. We have conducted ablation studies to evaluate the contribution attributable to input preprocessing using close-to-morphological segmentation, and also adjusted hyperparameters, and found that the segmentation is responsible for approximately 1.25% of the improvement.

In addition, we have presented a new dataset in Latvian, that contains dialogs between users and customer support specialists. The dataset in total has 283 dialogs with 688 user turns and 531 customer support representative turns, with each dialog containing no less than two user turns separated by a support representative turn, and with all user turns manually annotated for frustration intensity level. This dataset was used for training and assessing the reference neural network-based model for frustration prediction and its improved version.

Acknowledgments

The research has been supported by the European Regional Development Fund within the joint project of SIA TILDE and University of Latvia “Multilingual Artificial Intelligence Based Human Computer Interaction” No.1.1.1/18/A/148.

References


Yao, Yuanlin, Shuwei Wang, Ruifeng Xu, Bin Liu, Lin Gui, Qin Lu, and Xiaolong Wang. "The construction of an emotion annotated corpus on microblog text."


System Combination for Grammatical Error Correction
Based on Integer Programming

Ruixi Lin Hwee Tou Ng
Department of Computer Science
National University of Singapore
{ruixi,nght}@comp.nus.edu.sg

Abstract

In this paper, we propose a system combination method for grammatical error correction (GEC), based on nonlinear integer programming (IP). Our method optimizes a novel $F$ score objective based on error types, and combines multiple end-to-end GEC systems. The proposed IP approach optimizes the selection of a single best system for each grammatical error type present in the data. Experiments of the IP approach on combining state-of-the-art standalone GEC systems show that the combined system outperforms all standalone systems. It improves $F_{0.5}$ score by 3.61% when combining the two best participating systems in the BEA 2019 shared task, and achieves $F_{0.5}$ score of 73.08%. We also perform experiments to compare our IP approach with another state-of-the-art system combination method for GEC, demonstrating IP’s competitive combination capability.

1 Introduction

Grammatical Error Correction (GEC) is the task of detecting and correcting grammatical errors of all types present in sentences in an essay, and generating a corrected essay (Ng et al., 2014).

Most of the latest GEC systems rely on pre-training with synthetic data and fine-tuning with task-specific data, and employ deep neural networks with attention mechanisms (Bryant et al., 2019). Single GEC systems can be highly effective in capturing a wide range of grammatical error types (Ng et al., 2014), but each individual system differs in its strengths and weaknesses in correcting certain error types, and the differences could result from the synthetic data used in pre-training a GEC system (Bryant et al., 2019). Two main categories of synthetic data generation approaches have been introduced, including directly injecting noise into grammatically correct sentences according to error distributions (Zhao et al., 2019; Choe et al., 2019), or by back-translation (Sennrich et al., 2016; Xie et al., 2018; Kiyono et al., 2019). While both categories could help a system achieve a high recall across many error types, it is hard to obtain a single uniform GEC system that is good at correcting all error types.

Presented with this difficulty and the strengths of individual systems, combining single GEC systems is thus a promising and efficient way to further improve precision and recall. Creating an ensemble of multiple systems is a common approach when it comes to combining multiple models, and the work of Junczys-Dowmunt et al. (2018) has shown its effectiveness when combining single GEC models with different random initializations and configurations. However, this mode of combination requires altering the component systems to achieve a tight integration.

In contrast, we focus on combination methods that need to consider only the outputs of individual systems. State-of-the-art system combination approaches working in this manner include Susanto et al. (2014) and Kantor et al. (2019). The approach in Susanto et al. (2014) adopts the MEMT system combination technique of Heafield and Lavie (2010) and learns a combined corrected sentence which is made up of different parts of multiple system outputs. The work by Kantor et al. (2019) has proposed a system combination approach based on convex optimization. It treats single GEC systems as black boxes, rounding system weights to 0 or 1, and iteratively combines two systems at a time.

In this paper, we propose a novel system combination method based on nonlinear integer programming (IP). Our method optimizes an $F$ score objective based on error types, and combines multiple component GEC systems simultaneously with binary selection variables. In Section 2, we present related work on system combination for GEC. Then, our proposed IP approach is described in detail in Section 3. We present experimental setup and
results in Section 4, provide analysis of results in Section 5, and conclude in Section 6. Our source code is available at https://github.com/nusnlp/gec_ip.

2 Related Work

System combination approaches for GEC are based on pipelining, confusion networks, error types, or optimization. Pipelining approaches, such as the CAMB system (Felice et al., 2014), adopt a pipeline of simpler to more complex GEC systems for correction, but they suffer from error propagation. Confusion networks, especially the MEMT approach for GEC (Susanto et al., 2014), learn the optimal word choice at a sentence location via a decoding scheme. The error type-based approaches aim at edit selection per error type and per system. The LAIX system (Li et al., 2019) employs a confidence table and a rule-based conflict solver to select the optimal edits from component systems. Past work using integer linear programming (ILP) for GEC includes (Rozovskaya and Roth, 2013; Wu and Ng, 2013). Moving on to using optimization-based selection variables, the IBM approach (Kantor et al., 2019) combines edits for a pair of systems at a time, based on error types and subsets of corrections. Continuous selection variables are learned by maximizing the subset-based \( F_{0.5} \) objective. The IBM approach is most related to our proposed IP approach, and we highlight the key differences in Section 3.3.

3 System Combination

We combine systems based on the strengths of individual systems in terms of error types, and optimize directly the \( F \) score evaluation metric (van Rijsbergen, 1974) to obtain error type-based selection variables for each system. Compared to Kantor et al. (2019), we make several major changes to achieve good precision and recall while making the combination more efficient. An overview of our proposed IP approach is illustrated in Figure 1.

3.1 An Integer Programming-Based Approach

First, we observe that the selection variables learned by convex optimization in Kantor et al. (2019) are rounded to their nearest integers, either 0 or 1, for simplicity. This approximation of continuous variables raises the question of why binary variables were not directly used in the first place. A more direct solution is to adopt binary variables. Let

\[
x_{ij} = \begin{cases} 
1 & \text{if } S_i \text{ is used to correct } T_j \\
0 & \text{otherwise}
\end{cases}
\]

where \( S_i \) refers to system \( i \) in a set of \( M \) systems \( S = \{ S_1, \ldots, S_M \} \) and \( T_j \) refers to error type \( j \) in a set of \( N \) error types \( T = \{ T_1, \ldots, T_N \} \). Taking \( F \) score, the evaluation metric of GEC systems, as our objective function to maximize, we can formulate the GEC system combination problem as a nonlinear 0-1 integer programming (IP) problem as follows:

\[
\max F_\alpha(X) = \frac{(1 + \alpha^2) \cdot TP_{sum}}{(1 + \alpha^2) \cdot TP_{sum} + FP_{sum} + \alpha^2 \cdot FN_{sum}}
\]

s.t.

\[
\sum_{i \in S} x_{ij} = 1, \forall j \in T \tag{1}
\]

\[
x_{ij} \in \{0,1\}, \forall i \in S, j \in T \tag{2}
\]

where

\[
TP_{sum} = \sum_{i \in S} \sum_{j \in T} \lambda_{ij}^{TP} x_{ij} \tag{3}
\]

\[
FP_{sum} = \sum_{i \in S} \sum_{j \in T} \lambda_{ij}^{FP} x_{ij} \tag{4}
\]

\[
FN_{sum} = \sum_{i \in S} \sum_{j \in T} \lambda_{ij}^{FN} x_{ij} \tag{5}
\]

Equation (1) imposes the constraint that each error type is corrected by exactly one system. Equation (2) is the integer constraint, resulting in a 0-1 integer programming model. In Equations (3), (4), and (5), \( \lambda_{ij}^{TP} \), \( \lambda_{ij}^{FP} \), and \( \lambda_{ij}^{FN} \) respectively denote the true positive count, false positive count, and false negative count for system \( i \) and error type \( j \). In this paper, we set \( \alpha = 0.5 \) and optimize \( F_{0.5} \), the standard evaluation metric in GEC.

Moreover, the combination method in Kantor et al. (2019) uses the intersection of the edits of multiple systems, which can be too sparse to be useful when many systems are combined. Their iterative combination approach may alleviate the sparsity problem, but high computational cost is incurred when the number of component systems is large, due to the inherent combinatorial explosion of finding the optimal order of combination. In contrast, our approach requires no subset splitting, and optimizes all component systems simultaneously as indicated in Equations (3), (4), and (5).
3.2 Combination Procedure

The input to the IP approach is $M$ corrected sentences (of the same ungrammatical sentence) given by GEC systems $S_1, \ldots, S_M$. The output is a corrected sentence, after applying edit selection. The combination procedure consists of an optimization step followed by a correction (inference) step. Details are as follows.

**Optimization Step.** We compute the true positive, false positive, and false negative counts for each error type and component system on a training dataset. Utilizing these counts, the 0-1 integer programming model defined in Section 3.1 is solved using the commercial optimization software LINGO10.0\(^1\) to compute the optimal solutions for $x_{ij}$. In LINGO10.0, we adopt the INLP (integer nonlinear programming) model. For the experiments in this paper, the runtime for the LINGO solver to compute an optimal solution is 2 to 10 seconds.

**Correction Step.** During correction (inference), the system applies an edit (by system $i$ to correct an error of type $j$) to an input sentence if $x_{ij} = 1$ as determined by the LINGO solver. Conflicts of candidate edits from multiple systems (under different error types) can occur in the same location in a sentence. In other words, although a single system is used for each error type, a conflict can occur when two systems perceive an error in the same location to be of different error types, causing the location to have multiple candidate edits. When this happens, we set IP to randomly choose a candidate edit.

3.3 Key Differences from the IBM approach

Since the IBM approach (Kantor et al., 2019) is the most related work, we summarize the key differences of our IP approach from the IBM approach.

1. The IP approach directly combines all systems at once, as opposed to iteratively combining two systems at a time in the IBM approach, where the order of combination affects the outcome and there is a need to search for the best order of combination. In contrast, our approach avoids the problem of searching for the best order of combination.

2. Binary (0-1) integer selection variables are directly used, in contrast to approximation by integers in the IBM approach.

3. In the IP approach, we avoid having to perform subset splitting during optimization, in contrast to the IBM approach. For subset splitting, corrections from two systems are split into an intersection subset and subsets containing per system-only corrections, which can result in data sparsity.

\(^1\)https://www.lindo.com/index.php/products/lingo-and-optimization-modeling
4 Experiments

4.1 Component Systems

To apply the IP combination method on GEC, we choose three state-of-the-art GEC systems as component systems. They are the systems UEdin-MS (Grundkiewicz et al., 2019) and Kakao (Choe et al., 2019) (the top two systems from the restricted track of the BEA 2019 shared task), as well as Tohoku (Kiyono et al., 2019).

The three GEC systems share interesting commonalities and exhibit salient differences. The major commonalities are the use of the Transformer Big architecture (Vaswani et al., 2017) (the Kakao system uses a variant, the copy-augmented Transformer (Zhao et al., 2019)), pre-training on 40M to 100M synthetic parallel data, ensemble of multiple models, and re-ranking. They differ in the synthetic data generation methods, the monolingual sources, the implementation of the architecture, and re-ranking features. These characteristics lead to individual strengths in correcting different types of errors, allowing room for improvements via system combination.

4.2 Training and Evaluation

We use the official BEA 2019 shared task datasets for training and evaluation. We learn our selection variable values based on the output sentences of the component systems on the official validation set. At inference time, we apply combination on the output sentences of the component systems on the official blind test set of BEA 2019. The resulting output sentences are sent to the shared task leaderboard for evaluation, where $P$, $R$, and $F_{0.5}$ scores are the evaluation metrics. The official scorer is the ERRANT evaluation toolkit v2.0.0 (Bryant et al., 2017).

4.3 Experimental Results

Besides our proposed IP method, we compare with the MEMT-based system combination approach for GEC. We follow the work of Susanto et al. (2014) to use the open source MEMT toolkit (Heafield and Lavie, 2010) for experiments.

MEMT system combination performs two major steps to combine edits: pairwise alignment and confusion network decoding with feature weight tuning. Pairwise alignment is first performed using METEOR (Banerjee and Lavie, 2005) to form a search space for combination. The alignment recognizes exact matches, words with the same stem, synonyms defined in WordNet, and unigram paraphrases. Then a confusion network is formed on top of the aligned sentences and beam search decoding is used to form hypotheses. During beam search, scoring of partial hypotheses is performed by a set of features, including hypothesis length to be normalized, log probability from a language model, n-gram backoff from the language model, and matched n-grams between the sentences generated by the component systems and the hypothesis. Tuning of feature weights is performed using ZMERT (Och, 2003), optimized for the BLEU metric. We report the average score of three runs of each MEMT combination.

The scores for the component systems and the IP combination approach are reported in Table 1, and the scores for the MEMT approach are reported in Table 2. All combination scores using the IP method are higher than the individual systems’ scores. The best IP score is achieved by combining UEdin-MS and Kakao (1+2), and the $F_{0.5}$ score is 73.08%, which is 3.61% higher than that of the best individual component system UEdin-MS. Comparing IP with MEMT, the average $F_{0.5}$ score across all combinations of IP is 72.36%, and that for MEMT is 71.42%, so the average $F_{0.5}$ score of IP is 0.94% higher than that of MEMT. The IBM approach (Kantor et al., 2019) reported $F_{0.5}$ score of 73.18% when combining the component systems UEdin-MS and Kakao (1+2). Overall, the performance of IP is thus comparable to other state-of-the-art combination approaches.

<table>
<thead>
<tr>
<th>System</th>
<th>P</th>
<th>R</th>
<th>$F_{0.5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>UEdin-MS</td>
<td>72.28</td>
<td>60.12</td>
<td>69.47</td>
</tr>
<tr>
<td>Kakao</td>
<td>75.19</td>
<td>51.91</td>
<td>69.00</td>
</tr>
<tr>
<td>Tohoku</td>
<td>74.71</td>
<td>56.67</td>
<td>70.24</td>
</tr>
<tr>
<td><strong>IP</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1: 1+2</td>
<td><strong>78.20</strong></td>
<td>57.90</td>
<td><strong>73.08</strong></td>
</tr>
<tr>
<td>C2: 1+3</td>
<td>76.08</td>
<td>58.81</td>
<td>71.86</td>
</tr>
<tr>
<td>C3: 2+3</td>
<td>76.95</td>
<td>55.54</td>
<td>71.44</td>
</tr>
<tr>
<td>C4: 1+2+3</td>
<td>78.17</td>
<td>57.88</td>
<td>73.05</td>
</tr>
</tbody>
</table>

Table 1: Scores (%) of the component systems and the IP combination approach on the BEA 2019 blind test set.
## 5 Analysis of Results

We analyze how much the combined system using the IP approach improves over individual component systems, on a per-sentence basis. Since reference edits are unavailable for the blind test set, we base our analysis on the BEA 2019 validation set and split it into two halves: the first half for training and the second half for testing. For each input sentence, we compare the $F_{0.5}$ scores of its output sentences generated by Kakao, UEdin-MS, and the system obtained by IP combination of both. We assign each output sentence $s$ into one of two classes, based on whether (1) the $F_{0.5}$ scores of $s$ are identical in Kakao and UEdin-MS; or (2) the $F_{0.5}$ scores of $s$ are different.

The findings are summarized as follows. Of the 2,192 test sentences, there are 1,503 sentences where Kakao and UEdin-MS have the same $F_{0.5}$ score. For these sentences, the IP approach achieves the same or higher $F_{0.5}$ score on 1,501 sentences. The $F_{0.5}$ score of IP on these 1,501 sentences is 0.4% higher than each individual system. For the remaining 689 sentences that either Kakao or UEdin-MS performs better, 503 out of 689 sentences benefit from IP combination, with an increase of 11.58% in the overall $F_{0.5}$ score on the 503 sentences compared to the average $F_{0.5}$ score of Kakao and UEdin-MS. This analysis shows that the component systems benefit from the IP combination approach on a per-sentence basis.

## 6 Conclusion

In this paper, we have proposed a system combination approach for GEC based on nonlinear integer programming, which combines all systems at once. The use of binary selection variables is simpler and more direct, compared to using continuous variables then rounding them. The best $F_{0.5}$ score achieved is 73.08% on the BEA 2019 test set.

---

### Table 2: Scores (%) of the MEMT approach on the BEA 2019 blind test set.

<table>
<thead>
<tr>
<th>System</th>
<th>P</th>
<th>R</th>
<th>$F_{0.5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1: 1+2</td>
<td>72.52</td>
<td>60.92</td>
<td>69.90</td>
</tr>
<tr>
<td>C2: 1+3</td>
<td>73.06</td>
<td>60.75</td>
<td>70.29</td>
</tr>
<tr>
<td>C3: 2+3</td>
<td>75.84</td>
<td>58.28</td>
<td>71.50</td>
</tr>
<tr>
<td>C4: 1+2+3</td>
<td>79.17</td>
<td>58.68</td>
<td>73.98</td>
</tr>
</tbody>
</table>

Acknowledgements

This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-RP-2019-014). The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

### References


Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a


Multilingual Learning for Mild Cognitive Impairment Screening from a Clinical Speech Task

Hali Lindsay¹, Philipp Müller¹, Insa Kröger¹, Johannes Tröger¹ ², Nicklas Linz², Alexandra König³ ⁴, Radia Zeghari³, Frans RJ Verhey⁵, Inez HGB Ramakers⁵

¹German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
²ki-elements, Saarbrücken, Germany, 66111
³CoBTeK (Cognition Behaviour Technology) Research Lab, University Côte d’Azur, France
⁴National Institute for Research in Digital Science and Technology (INRIA), Nice, France
⁵Maastricht University, Department of Psychiatry & Neuropsychology, School for Mental Health and Neuroscience, Alzheimer Center Limburg, Maastricht, Netherlands, 6200

hali.lindsay@dfki.de, philipp.mueller@dfki.de, insak@coli.uni-saarland.de, johannes.troeger@dfki.de, nicklaslinz@googlemail.com, akonig03@gmail.com, Radia.zeghari@univ-cotedazur.fr, f.verhey@maastrichtuniversity.nl, i.ramakers@maastrichtuniversity.nl

Abstract

The Semantic Verbal Fluency Task (SVF) is an efficient and minimally invasive speech-based screening tool for Mild Cognitive Impairment (MCI). In the SVF, testees have to produce as many words for a given semantic category as possible within 60 seconds. State-of-the-art approaches for automatic evaluation of the SVF employ word embeddings to analyze semantic similarities in these word sequences. While these approaches have proven promising in a variety of test languages, the small amount of data available for any given language limits the performance. In this paper, we for the first time investigate multilingual learning approaches for MCI classification from the SVF in order to combat data scarcity. To allow for cross-language generalisation, these approaches either rely on translation to a shared language, or make use of several distinct word embeddings. In evaluations on a multilingual corpus of older French, Dutch, and German participants (Controls=66, MCI=66), we show that our multilingual approaches clearly improve over single-language baselines.

1 Introduction

Mild Cognitive Impairment (MCI) is a medical condition (Petersen et al., 2014) that often precedes Alzheimer’s Disease (AD). The development of cost-effective and scaleable screening approaches for MCI is crucial for the early treatment and management of AD (Dubois et al., 2016). The Semantic Verbal Fluency Task (SVF) is a promising screening approach as it combines a time-efficient testing procedure with the possibility of remote and automatic evaluation (Tröger et al., 2018). In this task, the testee is asked to name as many words as possible from a given semantic category (e.g. animals) in a given time (e.g. 60-seconds). Traditionally, the number of named within-category items is used to detect cognitive impairment. However, recent research has shown that in-depth analysis of the underlying cognitive strategies used for the SVF (e.g. semantic memory retrieval, executive control) enables a more fine-grained differential diagnosis (Tröger et al., 2019).

To harness the diagnostic power of the SVF, current automatic evaluation approaches identify semantic clusters in the participants’ word sequences, based on semantic word embeddings (Woods et al., 2016; Linz et al., 2017b; Paula et al., 2018). As the word embeddings used in these approaches are language-specific, training diagnostic machine learning approaches for target languages with small available datasets of SVF tests is challenging. Despite the potential of improving MCI classification by training on larger, multilingual data, all existing approaches for automatic MCI diagnosis are trained and evaluated on data from a single language.
In this paper, we for the first time investigate multilingual learning approaches for MCI screening from the SVF. To train a joint model that generalises across test languages we evaluate two approaches: (1) translation to a common language, and (2) the application of several distinct embedding resources to the same SVF productions. In line with the state of the art (Paula et al., 2018), we evaluate qualitative embedding-based approaches through an extrinsic quantitative downstream NLP application (Wang et al., 2018): classification between controls (HC) versus MCI from qualitative SVF features. In evaluations on French, Dutch, and German corpora we show clear improvements of the multilingual learning approaches over the single-language baselines. Our results show that the the performance of classical single-language, single-embedding approaches heavily depends on the combination of embedding and language, hindering generalizability. In contrast, by extracting features from several embeddings simultaneously and training over several languages, we achieve improved and more consistent classification performances across several test languages.

2 Related Work

Our work is related to clinical Evaluation, semantic word embeddings, as well as the automatic qualitative evaluation of verbal fluency tasks.

2.1 Clinical Evaluation

During an SVF trial, a person is asked to name as many words from a semantic category (e.g. animals) as they can in one minute. The person’s response is then scored as the number of unique words named excluding any repetitions. Typically, this word count is then used to determine if the person shows signs of cognitive impairment.

In addition to the word count, qualitative measures to evaluate underlying strategy—clustering and switching—have been proposed (Troyer et al., 1997). For this evaluation, consecutive words that have a discernible semantic relationship are considered to be in a cluster. For instance, in the SVF response “cat, dog, whale, dolphin...”, “Cat” and “dog” are common pets whereas “whale” and “dolphin” are marine mammals. The process of going from one cluster to the next is called switching.

Computing these additional metrics by hand is time-consuming and subjective. This has lead to developing automated methods of clustering and switching based on distributional semantics, or semantic word embeddings (Linz et al., 2017a; Clark et al., 2016).

2.2 Semantic Word Embeddings

Semantic word embeddings map words to a vector space encoding their semantic meaning. Words with high semantic similarity are mapped to vectors close in this semantic space, semantically dissimilar words to distant vectors. These semantic vectors are learned through a variety of algorithms on any large corpora of text with two main varieties of embeddings: contextual and non-contextual (Miaschi and Dell’Orletta, 2020). In a non-contextual word embedding, the vector representation is static, whereas, in a contextual embedding, the surrounding words are considered. For example, if we had 'cutting paper' and 'cutting class', a non-contextual word embedding would assign the same vector to 'cutting' in both phrases whereas a contextual embedding would take into account the difference of meaning.

Given the nature of the verbal fluency task, a non-contextual list of animals, this paper focuses on using different types of non-contextual word embeddings to investigate how to model a persistent underlying cognitive structure while combining data from multiple languages. To keep results comparable and reproducible, pre-trained publicly available models that are available in a range of languages are investigated namely, FastText (Bojanowski et al., 2016a), Spacy (Honnibal et al., 2020), and Wikipedia2Vec (Yamada et al., 2020a).

As semantic vectors are learned from large amounts of text corpora (usually Wikipedia and OSCAR common crawl), embedding quality heavily depends on the quantity of the available training data. While French, German and Dutch are relatively well-supported Indo-European languages, they are at a large disadvantage in comparison to English model resources. For instance, Wikipedia offers 6,317,662 articles for English but much fewer for French(2,337,481), German(2,586,965) or Dutch(2,058,488)1.

This presents a trade-off for approaching multilingual learning with semantic embeddings for clinical applications between maintaining the nuance of verbal fluency response in its native language or translating the response to English to take advantage of larger resources. In this paper, we

1https://en.wikipedia.org/wiki/List_of_WikipediasDetails_table
investigate both scenarios of multilingual machine learning for clinical models.

2.3 Automatic Qualitative Evaluation of VF Tasks

Verma and Howard (2012) showed that pathological semantic organization of speech is an effective proxy for underlying cognitive impairment in early AD—MCI. As a result, MCI screening from the SVF has leveraged a variety of computational models of semantic coherence across many languages. Early approaches for automatic semantic modeling of the SVF relied on classic co-occurrence measures for capturing AD-related semantic SVF markers (Clark et al., 2016; Pakhomov et al., 2012), graph-based measures (Lerner et al., 2009), or employed latent semantic analysis (Pakhomov and Hemmy, 2014; Pakhomov et al., 2015).

Most recently, semantic word embeddings have been used for automatic evaluation of verbal fluency tasks, including the SVF (Linz et al., 2017b; Paula et al., 2018; Kim et al., 2019; Lindsay et al., 2021a). For MCI screening, encouraging results were obtained with a variety of semantic NLP resources including word2vec (Linz et al., 2017b; König et al., 2018), WordNet (Paula et al., 2018), and Wikipedia backlink vector space models (Kim et al., 2019); Paula et al. (2018) and Linz et al. (2017a) reported classification performances of AUC 0.71 with a random forest classifier and F1 0.77 with a support vector machine, respectively.

While the type of embedding was found to significantly influence classification performance (Linz et al., 2017b; Paula et al., 2018), an approach combining different embedding types was not presented. Similarly, studied languages include French (Linz et al., 2017a), Korean (Kim et al., 2019), English (Pakhomov and Hemmy, 2014) and Brazilian Portuguese (Paula et al., 2018), but to our knowledge, no multilingual classification was investigated.

We argue that by extracting qualitative SVF features with multiple language-specific resources, we can train machine learning models across languages. Overcoming the issue of small clinical data sets and possibly building more robust models that generalize cognitive impairment that is not language-specific.

3 Methodology

3.1 Data

This study included SVF data from clinical datasets in three languages; French collected at Nice Institut Claude Pompidou Memory Clinic in France; German collected at the University Medical Centre Freiburg, Germany; and Dutch from Maastricht University Clinic, Netherlands. All participants performed a 60-second SVF for the category “animals” in their native language—in addition to a battery of cognitive tests—administered by a clinician. The recordings were manually corrected according to the CHAT protocol (MacWhinney, 1991; Karakostas et al., 2017; Tröger et al., 2017).

For all corpora, participants were excluded if they presented with comorbidities (e.g. apathy or depression). To control for confounding cognitive factors, samples from healthy controls (HC) and those with mild cognitive impairment (MCI) were matched for age and education in each language using the MatchIt package in R (Ho et al., 2011). The resulting demographic information for each corpus is listed in Table 1. A Wilcoxon non-parametric test is reported to check for differences in age and education between HC and MCI. All described studies were approved by national ethical committees and conform to the Declaration of Helsinki.

3.2 Embedding Resources

As the SVF does not evaluate language abilities but rather underlying processes of executive function and memory, we made use of non-contextual word embeddings. To keep results comparable and generalizable for future studies, we used pretrained mod-
Figure 1: Overview of our Multi-Embedding Multilingual Learning framework. The training data consists of SVF productions with MCI labels in different languages (French, German, Dutch). For each training sample, features are extracted using multiple embedding resources (FastText, Spacy, Wiki2Vec). With this training data, we learn a SVM classifier that is able to predict MCI versus HC at test time on SVF productions from any of the three languages.

3.3 Clustering-Based Features

The implementation for determining clusters using semantic embeddings followed Linz et al. (2017a). Each participant’s SVF production was transcribed and preprocessed into a sequence of only animal words represented by $a_1, ..., a_n$. A base threshold $T_p$ is determined for each participant $p$ by averaging the semantic similarity between all pairs of animal words in $p$’s production.

$$T_p = \frac{1}{n(n-1)} \sum_{i,j=1 \ldots n, i \neq j} sim(a_i, a_j)$$

Semantic similarity $sim$ was measured by the cosine distance between semantic embedding vectors $e_i$ extracted from words $a_i$, i.e. $sim(a_i, a_j) = \cos(e_i, e_j)$. Clusters were determined by comparing the semantic similarity of consecutive words $sim(a_i, a_{i+1})$ in the production to $T_p$. If the consecutive words were more similar than the base similarity threshold they were considered to belong to the same cluster. If the consecutive words were less similar than the base similarity threshold they introduced a cluster boundary, also referred to as a switch.

Based on the clusters obtained from a given participant with a given embedding, we computed the following features based on Linz et al. (2017a):

- **Mean cluster size** computed as the average number of words in a cluster.
- **Number of switches** calculated as the number of clusters minus 1.
- **Mean cluster distance** computed as the average semantic distance between all words in a cluster.
- **Mean switch distance** as the average semantic distance between centroids of adjacent clusters.

3.4 Multilingual Approaches

To combine multilingual data, we investigate two approaches. Section 3.4.1 proposes a method using available language-specific resources for each language and the section 3.4.2 translates all of the data to a common language, English.
3.4.1 Untranslated Multilingual, Multi-Embedding Approach
The central idea underlying the untranslated multilingual, multi-embedding approach is to maximize the available clinical data by using generalizable semantic features that robustly model cognitive impairment. To mitigate the possibility of fluctuating performance between language and embedding type, we propose using multiple embeddings for untranslated, multilingual data.

3.4.2 Translated to Common Language Approach
An alternative way to make use of training data from several source languages is to translate all SVF productions to a common language prior to feature extraction. We follow the methodology in (Paula et al., 2018) and translate all SVF productions to English before extracting word embeddings. For translation we first used Google translation API
\(^2\) and then manually checked and post-edited any words where the source word was identical to the target word. Due to privacy restrictions on medical data, a set of all mentioned animal names was extracted from the transcripts and a look-up dictionary was created mapping the animals of each language to its English equivalent.

3.5 Classification Experiments
From each of the French, Dutch and German productions as well as their English translations, the four described clustering-based features are extracted using each respective embedding resource.

3.5.1 Multilingual, Multi-embedding (ML-ME)
Figure 1 gives an overview of the multilingual, multi-embedding framework. For both the untranslated and translated approaches, the four features of underlying cognition are extracted. Each of the features vectors are concatenated into a single feature vector. This is the new representation of the SVF production that is then used to train the model and predict a label of HC or MCI.

3.5.2 Baseline Comparisons
Single Language, Single Embedding (SL-SE)
To test how well each embedding resource models each language, we trained on each combination of language and embedding resource individually.

Multilingual, Single Embedding (ML-SE)
To investigate how each embedding resource behaves in a multilingual training scenario, we trained a separate model for each embedding resource using all the language corpora.

3.5.3 Out-of-Vocabulary (OOV) Rate
In addition to the classification experiments, the out of vocabulary rate for each language for each embedding is considered. This is used as a quality control test to ensure words are not being dropped from the transcript when the features are being computed. The OOV rate is calculated as the unique number of word that are not in the semantic model divided by total produced animal words.

3.6 Evaluation
In line with previous work (König et al., 2018), classification was performed by a Support Vector Machine (SVM) with Radial Basis Function kernel implemented in sci-kit learn\(^3\) (Pedregosa et al., 2011), using default parameters for \(\gamma\) and \(C\). To maximize the amount of available data, testing for each model was done via leave-one-out cross-validation. Model performance was measured as area under the receiver operator curve (AUC). In the multilingual cases, a language-specific AUC was reported, where the multilingual model is evaluated separately on each target language. To compare the multilingual methods to the other approaches, AUC scores were averaged across the languages.

To nullify the effects of random initialization of the SVM optimization, we averaged AUC values obtained from 50 random initializations. To further test quality of the word embeddings, the rate of out of vocabulary words (OOV Rate) was reported as the percentage of words that did not have an embedding vector in the specific model.

4 Results
All results described in Section 3.6 are in Table 2 and Table 3. Table 2 displays the OOV rate analysis and Table 3 contains classification results.

A baseline was created using single language, single embedding (SL-SE) classifications. In the untranslated approach, Dutch had the lowest average across embeddings with an average AUC of 0.29, then French with 0.58, and finally German with 0.55. Similarly, Dutch had the lowest value (average AUC=0.38) across resources in the

\(^2\)https://cloud.google.com/translate

\(^3\)sci-kit learn version 0.24.0 for Python 3.7
Table 2: Out of Vocabulary Rate across embeddings

<table>
<thead>
<tr>
<th>Embedding</th>
<th>French</th>
<th>German</th>
<th>Dutch</th>
<th>English</th>
<th>Lang AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>OOV Rate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FastText</td>
<td>0.0</td>
<td>0.04</td>
<td>0.0</td>
<td>0.0</td>
<td>0.01</td>
</tr>
<tr>
<td>Spacy</td>
<td>0.0</td>
<td>0.07</td>
<td>0.0</td>
<td>0.0</td>
<td>0.018</td>
</tr>
<tr>
<td>Wiki2Vec</td>
<td>0.0</td>
<td>0.001</td>
<td>0.0</td>
<td>0.0</td>
<td>≤ 0.001</td>
</tr>
</tbody>
</table>

Table 3: Averaged AUC results for the Multilingual, Multi-embedding model and Baseline approaches. Single Embedding (SE), Multi-embedding (ME), Single Language (SL), Multilingual (ML), Average (AVG), Out of Vocabulary (OOV). Cross-Language AVG is the average AUC performance for the values in the row.

translated approach, then German (0.41) and finally French (0.50). No single embedding type showed consistent best performances. In the untranslated approach, French and Dutch performed best with Spacy, and German performed best with FastText. In the translated approach, French and German achieved their best performance with English Wiki2Vec embeddings, whereas the Dutch data worked best with FastText embeddings. The overall finding from the SL-SE baseline showed that no single embedding type performed best over the setting.

In a next step, we combine the datasets to create a multilingual training scenario for each of the embedding types (ML-SE). In both approaches, every classification improves with the multilingual data with the exception of the French Wiki2Vec embeddings in the untranslated case. To make more meaningful comparisons to the SL-SE and ML-SE cases, we aggregate over the languages for each embedding type and report a cross-language average (shown in the table as Lang AVG). We then compare the cross language averages in the single language and multilingual scenarios. In both the untranslated and translated scenarios we see overall improvement. In the untranslated case, we see an average improvement of 12 AUC points, with the largest improvement coming from Wiki2Vec (16 AUC points). In the translated case, using the combined data, we see an average improvement of almost 15 AUC points.

In the case of the untranslated data, we see the largest overall improvement in the multilingual, multi-embedding scenario. Averaging over the cross language averages (Lang AVG) of the ML-SE scenario produces an AUC of 0.60, while the ML-ME average reaches an AUC of 0.66. For the ML-ME scenarios in the translated case, by averaging across the ML-SE models we achieve a comparable AUC of 0.58 which is outperformed by the ML-ME model (0.62). However, in the translated case, the best results are produced in the multilingual scenario using Wiki2Vec embeddings (0.64).

In addition to the classification analysis, we investigated the Out-of-Vocabulary (OOV) rate for each of the embeddings in each of the languages as a form of quality control. The results in Table 2 show that our embeddings were suited for the task. Overall, we had no OOV words for French, English and surprisingly Dutch. However, German animals seem to be lacking from models of equivalent size. This could also be due to language-specific differences in morphology.

5 Discussion

In this study we investigate two different methods of combining multilingual data to build clinical models to distinguish between healthy controls and early signs of Alzheimer’s Disease (MCI); untranslated and translated.

While the multi-embedding method is best for when data is kept multilingual, if the data is translated and no longer in need of multilingual re-
sources, a single embedding type did emerge with the best performance, Wiki2Vec. Given this, in the case where a common language (especially English) can be achieved, according to our data, it may be best to find and use one embedding type.

In addition, we found that embedding type does make a difference in classification performance. Therefore, caution should be used when deciding on semantic resources. For instance, in the untranslated case, if we were to build a multilingual model and only use Spacy emebeddings, we would have relatively good performing classifier in French and German but the Dutch classification would not exceed chance performance. While combining embeddings may not yield the best results for each individual language, it results in the most uniform improvement in a multilingual—versus translated—setting.

However, translating the data to English, drastically improved Dutch performance, specifically with the Wiki2Vec models. Speculatively, the improved overall performance of the English translations with Wiki2Vec could be due to the backlink model where relationships are modeled through linked wikipedia page, situating Wiki2Vec to be very useful in modelling these semantic relationships from a cognitive standpoint in verbal Fluency tasks. However, based on these results, it seems that these relationships are mainly found in the English Wiki2Vec model, most likely due to the large discrepancy in the amount of available training data between the languages.

Beyond just the brand of embedding, there are pros and cons that come with each the untranslated and translated approach. By translating the data to English, we introduce possible errors based on how the data is translated. In this study, we chose to combine an automatic approach with a manual post-editing step, making the translated approach not fully automatic. From a clinical perspective, we do not know if the previous work on cognition applies to data that is translated to another language and then assessed. However, from a computational standpoint, if a reliable translation service for the source language to English exists, using the monolith of English resources presents as a reliable and effective alternative.

There are many challenges that arise when trying to concatenate data from multiple sources, thus specific caution should be taken on how to model data that has health implications. Our investigation of the two approaches (untranslated and translated) shows that SVF speech data can be combined to achieve results comparable to previous models. As no unified benchmark exists for HC vs. MCI detection from the SVF, our results can only cautiously be compared with previous work. However, we noticed that our best results from our best models for French(0.66), German(0.68) and Dutch (0.69) are in line with reported AUC values for French (0.76 (König et al., 2018)) and Spanish (0.75 (Paula et al., 2018)). It is worth noting, that these results are achieved without using the overall word count, which typically the strongest indicator for MCI detection from the SVF task.

6 Conclusion

Using multilingual cognitive data in both a untranslated multilingual, multi-embedding approach and translated to a common language approach improved classification over single language baselines.

This is promising not only for the feasibility of increasing the size of small clinical datasets in quick and cost-effective way, but it also opens the door for methodology on how we can use multilingual data to build more robust understanding of underlying cognitive conditions (Lindsay et al., 2021b; Fraser et al., 2019).

Future work should look at exploratory analysis for the compatibility of features computed from translated transcripts in the current clinical understanding. This could present translation as a viable option for low-resource languages, or taking advantage of larger resources, while still presenting explainable clinical solutions and improved classification performance. While the languages in this study are all in the same language family, this methodology should be tested with data from different language families to test for robustness of the solution.

As such, our proposed methodology provides insight into the effect of NLP resources for classifications on cognition as well as a tentative solution to the problem of combining multiple clinical datasets. This addresses the issue of small clinical data sets as well as opens the door for building robust models of cognition for clinically actionable solutions leveraging multilingual data, paving the way towards reaching the societal goal of cost-effective early AD detection.
References


Serguei V.S. Pakhomov, Susan E. Marino, Sarah Banks, and Charles Bernick. 2015. Using Automatic Speech Recognition to Assess Spoken Responses to


Naturalness Evaluation of Natural Language Generation in Task-oriented Dialogues using BERT

Ye Liu1, Wolfgang Maier2, Wolfgang Minker3 and Stefan Ultes4
1,2,4Mercedes-Benz AG, Sindelfingen, Germany
3Ulm University, Ulm, Germany
1ye.y.liu@daimler.com
2,4{wolfgang.maier,stefan.ультес}@gmail.com
3wolfgang.minker@uni-ulm.de

Abstract
This paper presents an automatic method to evaluate the naturalness of natural language generation in dialogue systems. While this task was previously rendered through expensive and time-consuming human labor, we present this novel task of automatic naturalness evaluation of generated language. By fine-tuning the BERT model, our proposed naturalness evaluation method shows robust results and outperforms the baselines: support vector machines, bi-directional LSTMs, and BLEURT. In addition, the training speed and evaluation performance of naturalness model are improved by transfer learning from quality and informativeness linguistic knowledge.

1 Introduction
With the increasing popularity of virtual assistants such as Alexa or Siri, users have a higher demand for conducting natural conversations. They would like to chat with these assistants more naturally—maybe even like talking to a real human being. One of the key questions arising from this, though, is how to measure the naturalness of the generated language. In the past, native speakers judged the quality of the generated language by answering questions like “is this utterance natural?” or “could it have been produced by a native speaker?” to rate the naturalness (Novikova et al., 2016). However, this approach heavily depends on manual effort and is impractical for broader use. On the other hand, the widely used automatic metrics for evaluating language generation, like BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005), rely on word overlap mechanism and compare the generated sentence to one or more human-created reference sentences (Stent et al., 2005), but cannot directly reflect naturalness information.

Likert scale ratings are generally used in human evaluation. The Figure 1 shows an example of 6-point Likert scale rating on naturalness of a sentence. In general, the human evaluation items (naturalness, coherence, etc.) are hard explainable objectively and native speakers purely rely on their underlying criteria to judge the performance of generated language without any specified rules. To alleviate the manual effort previously needed, the goal of this paper is to render the task of assessing the naturalness of generated language as an automatic reference-based classification problem having each Likert scale rating representing one class. To our knowledge, we are the first to propose this task and present an approach evaluating naturalness of generation through fine-tuning a pre-trained Language Model (LM).

The contribution of this paper is two-fold: a pre-trained BERT model is fine-tuned to estimate the naturalness of a generated sentence based on a reference sentence. Three baselines are proposed for robust performance comparison: a support vector machine (SVM) baseline using bag-of-words (BoW) vectors for input representation, a bi-directional Long Short Term Memory (LSTM) neural network, and the pre-trained BLEURT model (Sellam et al., 2020). Second, based on the positive correlation between naturalness and other annotated information: quality and informativeness of the generated sentence, the proposed method is extended to leverage this additional information through transfer learning. And the learning speed and estimation performance of naturalness evaluation model is significantly increased.

The remainder of this paper is structured as fol-
Section 2 introduces the related works. In Section 3, our proposed approach for BERT-based naturalness estimation is presented. In Section 4, the experiment setups are covered, which include fine-tuning BERT, comparison against the baselines, and transfer learning. Section 5 describes the experiments results. The last Section 6 draws conclusions and outlines future work.

2 Related Work

With the development of Natural Language Generation (NLG) applications and their benchmark datasets, evaluation of NLG systems has become increasingly important. Generally, multiple automatic metrics are used in parallel to evaluate the performance of language generation, such as BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) or ROUGE (Lin, 2004). However, Chaganty et al. (2018) demonstrated that the existing automatic metrics have poor instance-level correlation with mean human judgment and that they assign bad scores to many good quality responses.

Since automatic metrics still fall short of replicating human decisions (Krahmer and Theune, 2010; Reiter, 2018), many NLG papers include some form of human evaluation. For example, Hashimoto et al. (2019) report that 20 out of 26 generation papers published at ACL2018 presented human evaluation results for showing their robust performance comparison. Celikyilmaz et al. (2020) and Gatt and Krahmer (2018) also highlighted that human evaluation is commonly viewed as the best reliable way to evaluate NLG systems, but come with many issues, such as costly and time consuming and human judgement bias. And more importantly, the evaluation results from human efforts are not always repeatable (Belz and Reiter, 2006). Dusek et al. (2017) previously attempted to predict quality ratings of generated language by using Recurrent Neural Network (RNN) with the help of the Meaning Representations (MRs) and showed promising performance. However, our work is more focused on naturalness evaluation and based on the gold reference. In order to relieve the human labor, we propose a reference-based method for naturalness evaluation by utilizing neural network to learn the complicated linguistic relationship from sentences. And this work can be easily extended to other human evaluation criteria (coherence, quality, etc...), if the corresponding human evaluation data is available.

In recent years, with huge success of pre-trained LMs (Devlin et al., 2019; Radford et al., 2018), many machine learned metrics for evaluating generation are proposed. Especially the pre-trained BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) shows its superiority in this field. Shimanaka et al. (2019) presented automatic machine translation evaluation by using BERT and achieved the best performance in segment-level metrics tasks on the WMT17 dataset for all to-English language pairs. Zhang et al. (2019) proposed an automatic evaluation metric for text generation based on pre-trained BERT contextual embeddings: BERTScore. BERTScore computes the similarity of two sentences as a sum of cosine similarities between their tokens’ embeddings. And Zhang et al. (2019) showed BERTScore correlates better with human judgments and provides stronger model selection performance than existing metrics. Sellam et al. (2020) presented BLEURT, which continually pre-trained BERT on synthetic data and then fine-tuned on task-specific ratings. And Sellam et al. (2020) demonstrated BLEURT can model human assessment with superior accuracy. Given the superiority of BERT, we also apply the pre-trained BERT for our proposed naturalness evaluation on generated language.

3 Naturalness Evaluation Using BERT

The task of estimation the naturalness of a generated sentence is framed as a classification task. Two sentences are used as input: the candidate sentence to be scored and a reference sentence. The naturalness score is derived through fine-tuning a pre-trained BERT (Devlin et al., 2019) model. The main architecture of BERT uses the encoder of a Transformer (Vaswani et al., 2017), which is an advanced encoder-decoder architecture leveraging the attention mechanism. Considering that the input may be sentence pairs in several tasks, technical innovation Next Sentence Prediction (NSP) helps BERT to learn the relationship between sentence pairs by receiving pairs of sentences as input and separating them with [SEP] token to learn predicting if the second sentence is the subsequent sentence in the original document. To do so, a [CLS] token is added at the beginning for every input to learn the meaning of the entire input. Exactly because of these specific characters, eleven NLP tasks
in Devlin et al. (2019) obtained new state-of-the-art results by fine-tuning BERT.

Figure 2 shows the fine-tuning structure for naturalness estimation with a candidate sentence (sys_ref) and a reference sentence (orig_ref) as input. Example for a sys_ref and orig_ref is shown in Table 1. In accordance to the NSP task, representing both sentences on the input side is achieved by separating them with the [SEP] token. As can be seen in Figure 2, an additional [CLS] token is inserted at the beginning of the first sentence. The final output of the [CLS] token, which is called pooled_output, forms a representation of the entire input. Then a linear layer with softmax activation is added to the top of pooled_output to predict the probability of sentence-level naturalness label. To our knowledge, we are the first to fine-tune BERT to learn the abstract naturalness linguistic information and demonstrate the robust performance of this method.

4 Experimental Setup

In this section, the experiment procedure including the pre-processing of the used data is introduced. First, BERT is fine-tuned for naturalness estimation and then compared it with the baselines. Furthermore, external knowledge is added demonstrating the impact on naturalness estimation through transfer learning.

4.1 Dataset Preprocessing

The dataset\(^1\) (Novikova et al., 2017) comprises textual dialog responses produced by three data-driven NLG systems over three different domains. The three NLG systems are respectively RNNLG\(^2\), TGen\(^3\) and LOLS\(^4\). The applied domains are SF Hotel and SF Restaurant (Wen et al., 2015).

$$\text{sys}_{\text{ref}}: \text{zuni cafe, is expensive.} \quad \text{orig}_{\text{ref}}: \text{how about zuni cafe, an expensive one?}$$

<table>
<thead>
<tr>
<th>judge</th>
<th>naturalness</th>
<th>quality</th>
<th>informativeness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 1: Example of the pre-processed data set.

which provide information about hotels and restaurants in San Francisco, and BAGEL (Mairesse et al., 2010) that provides information about restaurants in Cambridge. Human annotations on naturalness, quality, and informativeness were collected for each NLG-produced text on a 6-point Likert scale by asking the annotator “could the utterance have been produced by a native speaker?”, “How do you judge the overall quality of the utterance in terms of its grammatical correctness and fluency?”, “Does the utterance provide all the useful information from the meaning representation?”, respectively.

Table 1 shows an annotation example of the data. The sentence pair input comprises sys_ref and orig_ref. The sys_ref presents the output of each of the above-mentioned three NLG systems, one at a time, while orig_ref denotes human written references from the original data. The judges (1, 2, 3) represent the three human raters. The naturalness score is our target label. In addition to the naturalness score, the informativeness score and the quality score are utilised in transfer learning experiments to improve naturalness estimation as introduced in Section 4.4. To derive a single label for each sentence pair, the median of the individual annotations is used. The Table 2 shows the distribution of the final processed data, which includes 11,353 samples. And we randomly split the data into train/dev/test with 80%/10%/10%.

4.2 Fine-tuning BERT

During fine-tuning, the entire pre-trained BERT model is optimised end-to-end. The output of token [CLS]: the pooled_output, is further fed to a linear layer with softmax activation function with parameters $W \in \mathbb{R}^{K \times H}$, where $H$ is the dimension of the hidden state vectors and $K$ is the number of classes. In this paper, we applied English uncased BERT-Base model\(^5\), which has 12 layers, 768 hidden states and 12 heads, for the naturalness classification. So $H$ is 768 and $K$ is 6 in our experiment.

\(^{1}\)https://researchportal.hw.ac.uk/en/datasets/human-ratings-of-natural-language-generation-outputs

\(^{2}\)https://github.com/shawnwun/RNNLG

\(^{3}\)https://github.com/glampouras/LOLS

\(^{4}\)https://github.com/UFAL-DSG/tgen

\(^{5}\)https://tfhub.dev/google/bert_uncased_L-12_H-768_A-12/1

<table>
<thead>
<tr>
<th>naturalness</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>data size</td>
<td>394</td>
<td>373</td>
<td>670</td>
<td>2,185</td>
<td>3,062</td>
<td>4,669</td>
</tr>
<tr>
<td>total</td>
<td>11,353</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Distribution of naturalness labels of the data set with the majority label marked red.
Figure 2: Fine-tuning BERT architecture for naturalness estimation

All hyper-parameters are tuned to our dataset. The batch size is 256 and the number of epochs is 25. Adam (Kingma and Ba, 2014) is used for optimization to minimize the cross-entropy with an initial learning rate of 5e-3.

4.3 Baselines

As this is a novel task, there is no available existing baseline for us to compare. Hence, we apply the following three baselines for performance comparison in order to show the robustness of our proposed method.

**BoW + SVM:** We firstly introduce a SVM classifier using BoW representation as baseline. The SVM (Suykens and Vandewalle, 1999) is a discriminative classifier formally defined by a separating hyperplane and is widely used for classification task because of significant accuracy with less computation power. The BoW model (Zhang et al., 2010) is a text representation that counts the occurrence of words within a document. The BoW approach is very simple and flexible, and can be used for extracting word features from documents. These numerical BoW vector features are used as input to a SVM with linear kernel having hyper-parameters $C = 1.0$ and $\gamma$ set to ’auto’.

**Bi-LSTM:** We also introduce the bidirectional LSTM (Bi-LSTM) with one layer for naturalness evaluation as baseline. The Bi-LSTM layer has the same hidden size as fine-tuning BERT, i.e., 768, and the output is forward to one linear layer with softmax function for naturalness classification. We almost remain the same hyper-parameters setting as BERT for Bi-LSTM training, i.e., 256 batch size, 25 epochs and 5e-3 initial learning rate. We restrict the number of LSTM-layers to one because multiple layers resulted in very slow training speed, as the LSTM cannot be trained in parallel, and worse performance.

**BLEURT:** In order to establish robust performance comparison, we also apply a pre-trained model for naturalness evaluation as the third baseline: BLEURT (Sellam et al., 2020), which is a machine learned automatic metric for text generation. BLEURT continually pre-trained BERT (Devlin et al., 2019) with a large number of synthetic reference-candidate pairs on several lexical- and semantic-level supervision tasks and then fine-tuned on multiple human ratings. Sellam et al. (2020) published multiple versions of BLEURT in the official repository, which includes BLEURT-tiny, BLEURT-base and BLEURT-large. More information could be found in the link. (Sellam et al., 2020) demonstrated that BLEURT is much closer to human annotation and also recommended to fine-tune the pre-trained BLEURT for custom applications. Hence, we apply the BLEURT-tiny in this work for naturalness classification by adding one additional linear layer with softmax function. We also tried the recommended BLEURT-base model for naturalness evaluation, however, it directly resulted in worse performance in our case.

4.4 Fine-tuning BERT plus Transfer Learning

Analysing the correlation of the naturalness scores with the respective quality and informative scores using Spearman rank correlation coefficient (Hauke and Kossowski, 2011) shows the positive results. The spearman’s correlation between naturalness and quality is $\rho = 0.60$ and between naturalness and informativeness is $\rho = 0.45$. Hence, this positive correlation is further leveraged through transfer learning (Pan and Yang, 2009) using the same BERT-based setup. The procedure is as follows: first, the BERT model is fine-tuned to quality (or in-
formativeness, respectively), and then this already fine-tuned model is continually fine-tuned once more using the naturalness score as target.

5 Results and Discussion

The results of our proposed approach (BERT) for estimating the naturalness of a generated sentence given an additional reference are depicted in Table 3. In addition to the baselines: BoW + SVM and Bi-LSTM and BLEURT, the majority class accuracy is also shown. It is calculated as the proportion of the majority class (naturalness score 6) resulting in $\frac{4669}{11353} = 0.41$. BERT + TLI and BERT + TLQ represent the Transfer Learning results from Informativeness (TLI) and Quality (TLQ) respectively.

The results in Table 3 show that all BERT-based approaches outperform the baselines for classifying the naturalness of a generated sentence achieving a higher overall accuracy.

Even though the data set is imbalanced as shown in Table 2, Madabushi et al. (2019) indicate that BERT is capable of handling imbalanced classes with no additional data augmentation, which is also confirmed in our work. The Table 4 shows the accuracy of different naturalness score on our proposed model BERT with test data. Even if the data is seriously imbalanced, every naturalness class has comparative accuracy.

Given the imbalanced data set, the macro F1 score, recall and precision are also computed to show the robustness of our proposed approach. Moreover, through the transfer of quality (or informativeness) knowledge to naturalness training, the performance of naturalness estimation is further improved and training speed has also been greatly promoted. Figure 3 shows that the naturalness training based on transfer learning is faster and tends to be stable after only 5 epochs. Table 3 shows that transferring knowledge from quality results in the highest improvement on naturalness estimation. This is also consistent with the Spearman correlation of the naturalness scores and quality scores which is higher than the correlation of the naturalness scores and informativeness scores.

The Table 3 shows that using the BLEURT model for naturalness evaluation results in the worst performance even though BLEURT was already pre-trained on multiple tasks. The possible reason is that the BLEURT was pre-trained with multiple automatic metrics, hence, it has no superiority in our naturalness classification task.

6 Conclusion and Future Work

In this paper, we proposed a novel task of automatically estimating the naturalness for task-oriented generation based on a human reference. We proposed a robust estimation approach by fine-tuning the pre-trained BERT model which outperforms an SVM classifier, Bi-LSTM, fine-tuned BLEURT and majority class accuracy. Taking advantage of the positive correlation of naturalness on quality (or informativeness), we successfully improved the naturalness training speed and estimation performance through transfer learning.

This work sheds light on research towards naturalness evaluation by neural network learning. The final goal of our work is to relieve the human labors from naturalness evaluation task and realize the automatic naturalness evaluation for dialogue generation. Hence, we will firstly collect more human evaluation data for future work. Because the human evaluation data, which is already shared and public on the internet, is very limited. With more collected human evaluation data, we are also interested in the performance of our proposed method on other
<table>
<thead>
<tr>
<th></th>
<th>majority class</th>
<th>BLEURT</th>
<th>BoW + SVM</th>
<th>Bi-LSTM</th>
<th>BERT</th>
<th>BERT + TLI</th>
<th>BERT + TLQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1 score</td>
<td>-</td>
<td>0.13</td>
<td>0.66</td>
<td>0.69</td>
<td>0.83</td>
<td>0.84</td>
<td>0.86</td>
</tr>
<tr>
<td>recall</td>
<td>-</td>
<td>0.22</td>
<td>0.66</td>
<td>0.66</td>
<td>0.83</td>
<td>0.83</td>
<td>0.84</td>
</tr>
<tr>
<td>precision</td>
<td>-</td>
<td>0.18</td>
<td>0.67</td>
<td>0.73</td>
<td>0.82</td>
<td>0.85</td>
<td>0.89</td>
</tr>
<tr>
<td>accuracy</td>
<td>0.41</td>
<td>0.42</td>
<td>0.68</td>
<td>0.74</td>
<td>0.85</td>
<td>0.87</td>
<td>0.88</td>
</tr>
</tbody>
</table>

Table 3: Performance comparison of different methods shows the superiority of our proposed BERT on naturalness evaluation.

<table>
<thead>
<tr>
<th>naturalness</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>test size</td>
<td>41</td>
<td>35</td>
<td>65</td>
<td>227</td>
<td>305</td>
<td>462</td>
</tr>
<tr>
<td>prediction size</td>
<td>37</td>
<td>27</td>
<td>46</td>
<td>180</td>
<td>240</td>
<td>441</td>
</tr>
<tr>
<td>accuracy</td>
<td>0.90</td>
<td>0.77</td>
<td>0.71</td>
<td>0.79</td>
<td>0.79</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Table 4: Accuracy of different naturalness scores on BERT with test data.

evaluation criteria, such as quality, coherence etc. And we will further verify the performance of our proposed method on chit-chat and open domain dialogue generation.

References


Towards the application of calibrated Transformers to the unsupervised estimation of question difficulty from text

Ekaterina Loginova  
Ghent University, Ghent, Belgium  
ekaterina.loginova@ugent.be

Luca Benedetto  
Politecnico di Milano, Milan, Italy  
luca.benedetto@polimi.it

Dries Benoit  
Ghent University, Ghent, Belgium  
dries.benoit@ugent.be

Paolo Cremonesi  
Politecnico di Milano, Milan, Italy  
paolo.cremonesi@polimi.it

Abstract

Being able to accurately perform Question Difficulty Estimation (QDE) can improve the accuracy of students’ assessment and better their learning experience. Traditional approaches to QDE are either subjective or introduce a long delay before new questions can be used to assess students. Thus, recent work proposed machine learning-based approaches to overcome these limitations. They use questions of known difficulty to train models capable of inferring the difficulty of questions from their text. Once trained, they can be used to perform QDE of newly created questions. Existing approaches employ supervised models which are domain-dependent and require a large dataset of questions of known difficulty for training. Therefore, they cannot be used if such a dataset is not available (e.g. for new courses on an e-learning platform). In this work, we experiment with the possibility of performing QDE from text in an unsupervised manner. Specifically, we use the uncertainty of calibrated question answering models as a proxy of human-perceived difficulty. Our experiments show promising results, suggesting that model uncertainty could be successfully leveraged to perform QDE from text, reducing both costs and elapsed time.

1 Introduction

Question Difficulty Estimation (QDE), also known as “question calibration”, is a crucial task in education. In Computerized Adaptive Testing (Linden et al., 2000), for instance, students are shown questions that are suitable for their skill level. When a question is miscalibrated (i.e. its difficulty has been erroneously estimated), it can be either too hard or too easy for a student, which would negatively affect their learning outcome (Wang, 2014). If the questions are too hard, students might get frustrated and lose motivation; if they are too easy, students are not adequately challenged (Papousek et al., 2016). In either case, their learning experience is worse than if the questions were of appropriate difficulty, which, especially in the context of large-scale online courses, gives rise to the methods of automated difficulty estimation.

Traditionally, QDE is performed either i) manually (Attali et al., 2014) or ii) with pretesting (Lane et al., 2015). Manual calibration involves one (or more) human experts labelling the question by selecting a numerical value representing its difficulty, a method that is subjective and not scalable. Meanwhile, pretesting involves deploying the new question in an exam as if it was a standard exam question, but without using it to assess students and without telling them that there is a question under pretesting. The other questions of the exams are then used to actually assess the students, and their answers help to calibrate the question under pretesting. This approach indeed leads to an accurate calibration, but it introduces a long delay between the time of question creation and when the new question can be used to assess students. Besides, it requires the new questions to be shown to students before being actually used for scoring them, which is undesirable.

Recent studies tried to address the limitations of the traditional approaches by performing QDE with Natural Language Processing (NLP) techniques (Ha et al., 2019; Qiu et al., 2019; Benedetto et al., 2020b, 2021; Xue et al., 2020; Huang et al., 2017). They are all based upon the same general idea: starting from a set of calibrated questions, we train a supervised machine learning model to infer the difficulty of questions from their text. Once the model is trained, it is used to calibrate newly-generated questions, overcoming (or at least reducing) the need for pretesting and manual calibration. Although these techniques were proposed to enable an immediate calibration of new questions, they
have two major limitations due to their supervised manner: i) they require thousands of calibrated questions as a training set, and ii) they cannot perform cross-domain QDE. In other words, the training questions must assess the same topics as the new questions, which the model will later be used on, thus limiting its applicability (e.g. when introducing new courses in an e-learning platform). These limitations are intrinsic to such approaches and cannot be addressed by improving the accuracy of the models.

In this work, we explore an approach that is a total shift in the paradigm of QDE from text and which could potentially overcome both limitations. The intuition is to build an end-to-end Question Answering (QA) model that answers Multiple Choice Questions (MCQs) and use its uncertainty (which can be interpreted as representing the machine-perceived difficulty) as a proxy for human-perceived difficulty, which is the final target of the estimation. Previous research already hypothesized that there might be a relation between human-perceived difficulty and machine-perceived difficulty (Ha et al., 2019), but leveraged the machine-perceived difficulty as a feature for a supervised model, thus facing the same limitations as the approaches to QDE from text mentioned above. On the contrary, the approach we propose here performs QDE by leveraging the confidence of a trained QA model: specifically, we compute the variance of the probability distribution over the possible answer choices of the MCQ under calibration. Crucially, this approach is model agnostic and can be used on any QA model that outputs scores for the possible choices of an MCQ. Architecture-wise, in this study, we experiment with Transformers (Vaswani et al., 2017) for QA. In order to understand how well the proposed approach performs with different QA models, we use three different Transformers: BERT (Devlin et al., 2019), DistilBERT (Sanh et al., 2019), and XLNet (Yang et al., 2019). We choose them because they offer variety and differ in sizes, and were all demonstrated to perform well in several language understanding tasks.

However, large neural classification models tend to be overconfident in their predictions (Desai and Durrett, 2020), which might hinder the possibility of using their uncertainty as a proxy for question difficulty. In order to understand whether this is really an issue for unsupervised QDE from text and to explore possible solutions, we also experiment with calibrated QA models. Calibrating a model involves aligning the posterior probabilities with the empirical likelihoods (Guo et al., 2017). For instance, if we consider all the predictions for which a model has the confidence of 75%, the true accuracy must be 75% if the model is perfectly calibrated. We remark here that calibration and accuracy are not directly related: in fact, a model might be accurate but not calibrated or, on the other hand, not very accurate but well-calibrated. Calibration can be intuitively interpreted as the “awareness” of the model of its capabilities. In practice, several techniques can be used for calibrating neural models, and they will be discussed in Section 3. For simplicity and computational reasons, we use the ensembling technique.

We experiment on the large question-answering dataset RACE (Lai et al., 2017) to assess the QDE capabilities of the proposed approach. Specifically, we evaluate it on the task of pairwise difficulty prediction: given a pair of questions from RACE, the objective is to indicate which question of the pair is more difficult. As the gold standard for the difficulty, we use i) the difficulty level available in RACE and ii) additional human labels obtained with crowd-sourcing. The experimental results are promising and suggest that the proposed approach could be used to perform QDE from text in an unsupervised manner leveraging the uncertainty of QA models. We also show that choosing the underlying QA model is not straightforward, and the best results are obtained leveraging models that are both accurate and calibrated.

Our contributions are as follows: i) we propose an unsupervised way for QDE from text that does not require answer logs or question difficulty labels, only the text of the MCQ and the possible choices, ii) we experiment with modern Transformer-based architectures to demonstrate the viability of the proposed approach.

We share our code publicly1.

2 Related Work

The earliest research about QDE from text focused on MCQs, using bag-of-words and the similarities between question, correct choice, and distractors (incorrect choices) for the estimation (Alsubait et al., 2013; Ha and Yaneva, 2018; Kurdi et al., 2016; V and Puligundla, 2015). However, they are

1https://bit.ly/3b4tPLN
generally outperformed by the more recent models, which are based on machine learning techniques.

Ha et al. (2019) introduced a model to estimate from a question text its correctness, which is defined as the fraction of students who correctly answered the question. This model was trained using question texts and a large dataset of medical documents (i.e., books, papers). Similarly, the model proposed by Qiu et al. (2019) is trained on a dataset of medical documents and question texts to estimate the wrongness of newly generated questions. Benedetto et al. (2020b,a) proposed R2DE, a model that estimates the difficulty of newly generated MCQs, using as input only the text of the questions and the text of the possible choices, without any additional data. Xue et al. (2020) explored the effects of transfer learning for question calibration from text. Specifically, the authors fine-tune pre-trained ELMo embeddings (Peters et al., 2018) for the task of response time prediction and subsequently perform a second fine-tuning for the task of QDE. Lastly, Huang et al. (2017) propose a neural model for the estimation of the difficulty of reading comprehension questions.

From a high-level perspective, all these models are based on the same idea: the real question difficulty (obtained either with pretesting or manual calibration) is used as the target value for training a supervised machine learning model that performs QDE from text for newly generated questions. The downside of this approach is that it can work only as long as the new questions belong to the same domain as the training questions. Moreover, such models need a large number of calibrated questions for training, which might be too costly to obtain, especially for smaller institutions. (Wang et al., 2014) employs a pairwise difficulty comparison scheme similar to the one we will, but they still require the user responses for the algorithm to work, same as (Narayanan et al., 2017).

Motivated by this, in this work, we explore the possibility of completely shifting the paradigm for the task of QDE from text. The proposed approach leverages the uncertainty of a QA model as a proxy for question difficulty and uses this model-perceived difficulty to calibrate newly generated questions. This unsupervised approach does not require ground truth difficulty labels, but only the text of the questions and, in the case of MCQs, of the possible answer choices.

3 Calibration of Neural Models

Several techniques exist to calibrate neural network classifiers. The most popular ones are the following: i) vanilla: maximum softmax probability, which usually does not lead to calibrated classifiers (Hendrycks and Gimpel, 2017). ii) Temperature scaling: a posterior calibration technique using a validation set (Guo et al., 2017; Desai and Durrett, 2020). iii) Bayesian deep learning, which requires alterations to the training procedure and is computationally expensive. iv) Ensembles: consists in independently training $M$ models on the entire dataset using different random initializations (Lakshminarayanan et al., 2017) or dropout (Gal and Ghahramani, 2016; Srivastava et al., 2015) and averaging their predictions.

Focusing on pre-trained Transformers, previous research (Desai and Durrett, 2020) showed that pre-trained BERT is fairly well-calibrated for in-domain tasks, but it is miscalibrated for out-of-domain tasks. To the best of our knowledge, no previous research experimented with the calibration of DistilBERT and XLNet.

In this work, we use deep ensembles with $M$ equal to 3, since we observed that this approach led to fairly well-calibrated models for all the Transformer architectures under evaluation. Additionally, we also experimented with ensembles made of the combination of models with different architectures (e.g., BERT with XLNet, etc.).

Considering the metrics existing to evaluate model calibration, one of the most commonly used approaches is the Expected Calibration Error (ECE) (Naeini et al., 2015), which compares the confidence and the accuracy of the model. More precisely, it defines miscalibration as the difference in expectation between confidence and accuracy. Thus, ECE approximates the miscalibration by partitioning the predictions in a number $M$ of bins and averaging the difference between the accuracy and confidence obtained in each bin.

4 Models

The approach proposed in this paper leverages the confidence of a QA model to perform QDE for MCQs in an unsupervised manner. Specifically, it leverages the output scores, one for each possible answer choice, produced by the model for each question, and the only requirement is that such output scores represent a probability distribution (i.e., they sum to 1, such as the softmax scores of
a neural network). We experiment here on MCQ with four possible choices, but the approach can be easily scaled to MCQ with different numbers of answer choices. It is important to note that this approach to unsupervised QDE from text is “QA model agnostic”, in the sense that it does not need any information about the underlying model but only the scores produced by it. We leverage the softmax scores produced by the underlying QA model to measure the model-perceived question difficulty, which is considered a proxy for human-perceived difficulty. In practice, we convert the raw softmax scores of each question into a single numerical value by computing their variance. We assume that larger values of variance indicate easier questions (since it means that the model is more certain in the estimation).

We also experimented with some alternatives, but they were generally outperformed by using the score variance (although the difference was not major). Specifically, we also experimented with: i) keeping only the highest softmax score, and ii) computing the difference between the highest and the second-highest softmax score.

In order to understand how the proposed approach performs with softmax scores from different underlying QA models, we experiment with three Transformer models: i) BERT (Devlin et al., 2019), ii) DistilBERT (Sanh et al., 2019), and iii) XLNet (Yang et al., 2019). The details of each architecture are beyond the scope of this research; we refer to the original papers for their description. For our study, it is essential to know that they are all publicly available neural network models, which are pre-trained on several language understanding tasks, and that they can be fine-tuned for different downstream tasks with minimal changes to the architecture. Precisely, in this study, we add a multiple-choice classification layer on top of each original model, which is a common task in the literature. The three models have diverse sizes and architectures, even though DistilBERT is strongly related to BERT since it is obtained from it by using knowledge distillation (Hinton et al., 2015) to reduce the number of hidden layers.

We implement all the models with the HuggingFace transformers library (Wolf et al., 2019), using in all cases the pre-trained base-cased version. We fine-tune them using Google Colab GPUs. The parameter configuration for the QA models is taken from the literature; the specific values are shown in Table 1. The table also shows, for each model, the accuracy we obtain in the QA task on RACE, which is in line with previously reported results.

As suggested by (Lakshminarayanan et al., 2017), we ensemble models to reduce miscalibration. In practice, we proceed as follows. First, we i) train five instances for each architecture (i.e. BERT, DistilBERT, XLNet), and each instance is trained on the entire training dataset (randomly shuffled), with a different random initialization. Then, we ii) pick the three best performing instances of each architecture, considering the test accuracy on the QA task. Lastly, we iii) build the ensembles by averaging (separately for each test question) the softmax scores produced by the three instances of each architecture so that each of the four answer options is assigned a single score from 0 to 1. These scores indicate the probability (according to the model) of each option being correct. In addition to building an ensemble for each architecture, we also build “hybrid” ensembles in the same way but averaging the predictions of instances of different architectures.

## 5 Data

In this study, we use the reading comprehension RACE dataset\(^2\) and two datasets derived from it, which contain pairs of questions and a label indicating which question of the pair is more difficult. The entire RACE dataset is used to train the QA models, while the two other datasets are used for the experiments on QDE from text.

### 5.1 RACE

The original RACE dataset contains 25,000 passages in English from middle and high school reading comprehension exams, with four MCQ associated with each text (100,000 questions in total). All the questions are MCQ with four possible choices (see an example in Figure 1). The questions are

<table>
<thead>
<tr>
<th>Parameter</th>
<th>BERT</th>
<th>DistilBERT</th>
<th>XLNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>input text len.</td>
<td>256</td>
<td>512</td>
<td>256</td>
</tr>
<tr>
<td>learning rate</td>
<td>2e-5</td>
<td>5e-5</td>
<td>2e-5</td>
</tr>
<tr>
<td>Adam epsilon</td>
<td>1e-6</td>
<td>1e-8</td>
<td>1e-6</td>
</tr>
<tr>
<td>weight decay</td>
<td>0.05</td>
<td>0.05</td>
<td>0.01</td>
</tr>
<tr>
<td>n. epochs</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 1: Training configuration of the QA models.

\(^2\)www.cs.cmu.edu/~glai1/data/race/
designed to require more extensive reasoning skills than other QA datasets such as SQuaD (Rajpurkar et al., 2016) and differ in reasoning types required to answer them. For example, they cover passage summarization and attitude analysis, which means that the answer cannot always be extracted directly from the passage. As a result, neural methods have a significant performance gap compared to humans: 66.7% (with XLNet-base) and 94.5% accuracy, respectively. The questions can be separated into two types based on their syntax: interrogative or cloze. Interrogative questions are the ones that end with a question mark, while cloze questions contain a gap that has to be filled in.

5.2 PairRACE HM

For constructing this dataset, we use the level label available in RACE, which indicates the level of examination (high or middle) of each question. Specifically, we use level as an indication of question difficulty and prepare 2,062,096 pairs of question, such that each pair contains one middle question and one high question (related to different passages). This dataset is then used to evaluate the proposed approach in the task of pairwise difficulty estimation: basically, given a pair of question, we check whether the proposed approach labels the high question as being the more difficult one.

5.3 PairRACE CS

The level label does not contain a numerical estimation of question difficulty, and there is no way of knowing how much harder the high questions are. Therefore, we also build a dataset to evaluate how well the proposed approach performs at a more focused level difficulty estimation, which differentiates the difficulty of the question within the middle level and within the high level. Such fine-grained information about the difficulty is not available in RACE; thus we manually annotate a subset of the question pairs by crowd-sourcing on the Amazon Mechanical Turk platform.

First, 80 pairs of questions from the test set were randomly chosen (both questions in each pair have the same level and correspond to the same passage, with 77 unique passages used in total). An annotator was then presented with a passage and a pair of questions, along with their answer options, and asked to identify the more difficult question. Each question was first labelled by two of the authors (with Cohen’s kappa of 0.30) and then passed to turkers. The only condition imposed on turkers was to be native English speakers. Each question was answered by one crowd-worker; thus, in combination with our labels, we obtained three labels per question pair. To encourage a thoughtful approach, we added an obligatory field in which we asked the turkers to provide a brief motivation for their choice. An example reasoning is the following: “Question 1 requires you to think beyond the passage content, to extrapolate and predict the next step, while 2 just asks to give a title to the content.” There were multiple recurring indicators of how humans estimate difficulty. As expected, the questions which require searching for a named entity or finding a simple fact statement are considered simpler, while summarising information or giving a title is more challenging. The text’s location contributes as well – it is easier to answer the question if the cue can be found in the first or the last sentences of the passage. Ultimately, the Fleiss’ kappa agreement was 0.21, which is “fair”, according to guidelines by (Landis and Koch, 1977). Still, we recognise the possible issues with a relatively low agreement, and, in the experimental evaluation, we separately consider the performance on the question pairs with full agreement.

6 Experiments

The goal of our experiments is to perform unsupervised QDE from text, using only the softmax scores produced by QA models. More precisely, we do not estimate directly question difficulty but evaluate the proposed approach on the task of pairwise difficulty prediction: given a pair of questions, the task consists in identifying the one that is more difficult. The manual labels are obtained, also within the pairwise comparison framework. This way,
we investigate i) whether the machine uncertainty leads to a notion of difficulty that aligns well with the human one – which is represented by the level and the crowdsourced labels – and ii) whether it can be useful in practical applications when logs of answers or calibrated questions are not available for training.

### 6.1 Evaluating QA accuracy and calibration

Before actually evaluating the proposed approach on the task of pairwise difficulty prediction, we evaluate the QA accuracy and the calibration of the underlying models to explore the relations between these and the accuracy in the task of pairwise difficulty prediction. Results are shown in Table 2.

For QA, we use accuracy as a metric (the higher, the better), while for calibration, we use Expected Calibration Error (ECE, the lower, the better). It should be noted that ECE is used to diagnose whether the model’s confidence can be a reliable proxy for difficulty or not.

For each underlying model (i.e. BERT, DistilBERT, XLNet), we present the results for three single instances and the calibrated ensemble. The single instances are identified by a number, which is the random seed used during training. While the specific value of the random seed is not meaningful in itself for this analysis, we use it to distinguish between different instances of the same architecture. Ensembles are indicated by “E”.

As for the QA accuracy, we can see that, considering the same architecture (e.g. BERT) the accuracy of the ensembles is generally better than the single models, both on the eval dataset and the test dataset. However, this is not true for the “hybrid” ensembles, which perform worse than the model they are obtained from.

Similar results can also be seen for model calibration: ensembles generally have lower ECE, meaning that they are better calibrated. Indeed, BERT (E) has an ECE of 0.10, while the average of the single models is over 0.14, and XLNet (E) has an ECE of 0.11, the average of the single instances being 0.15. This trend is not as visible for DistilBERT, as the ensemble model has an ECE only slightly higher than the average of the single instances (0.07 compared to 0.06).

### 6.2 Pairwise difficulty prediction

Given as input a question pair, the unsupervised pairwise difficulty prediction task consists of predicting which question of the pair is more difficult. We evaluate the proposed approach using different underlying models (both single instances and calibrated ensembles), and compare it with three baselines based on ELMo (Peters et al., 2018). It should be noted that internally there is no difference indicated between comprehension and knowledge questions and each model is thus evaluated on the same subset.

i) ELMo\(_C\) (comprehension): we calculate the cosine distance between the ELMo embeddings of the question and of the passage; the question with the larger distance is labelled as more difficult.

ii) ELMo\(_K\) (knowledge): we calculate the average distance between the correct answer option and the distractors; the more difficult question has the lowest distance. This is a standard approach in the literature (Alsubait et al., 2013; Kurdi et al., 2016).

iii) ELMo\(_{QA}\): a QA model built upon ELMo. Given a passage and an MCQ, it selects the answer by picking the choice which has the lowest cosine distance to the passage. It produces one score for each possible choice, and we use our approach directly on these scores (after normalization).

As introduced in Section 4, the proposed ap-
Table 3 presents the results obtained on PairRACE_HM and PairRACE_CS, using accuracy as evaluation metric. Each row shows the results for a different model, and we can identify three groups: the baselines, the single instances, and the ensembles.

### 6.2.1 High vs middle (PairRACE_HM)

The columns on the left present the results obtained using the `level` label as ground truth difficulty. We separately present the accuracy i) on all the question pairs and ii) on the pairs in which both questions were Correctly Answered (CA) by the QA models; \( n \) is the number of question pairs.

Considering all the pairs, we can see that the proposed approach consistently performs at least as well as the baselines, both when using the single models and when using the ensembles, although the improvement is not major. The only exceptions are DistilBERT (3) and DistilBERT-XLNet (E).

Interestingly, if we compare Table 3 and Table 2, we can see that there is no clear correspondence between the accuracy in the QA task and the accuracy in pairwise difficulty prediction or between the ECE and the accuracy in pairwise difficulty prediction. For instance, DistilBERT (3) has the lowest (i.e. best) ECE, but it is outperformed by all the ensembles (except the hybrid ones) in the pairwise difficulty prediction task. This suggests that the calibration of the QA model is not the only factor to take into consideration when using its uncertainty for QDE and that its QA accuracy also has an important role.

Comparing the accuracy of the ensembles and the single models, we observe that there is no appar-
ent improvement in using calibrated ensembles. It is especially noticeable when considering only the questions that the models correctly answered. However, in a real-world unsupervised scenario, the true difficulty labels are not available for choosing the best performing model with cross-validation, and neither the accuracy in the QA task nor the ECE is sufficient to pick the best performing model. Therefore, we argue that the usage of calibrated ensembles is a better solution as it allows to avoid the oscillations of single instances (e.g. from 0.56 to 0.60 of the single DistilBERT models against the 0.58 of the ensemble). However, this is true only for ensembles of models with the same architecture. Hybrid ensembles did not lead to better performance; thus, we argue that they should not be used for the task of unsupervised QDE from text.

6.2.2 Crowdsourced labels (PairRACE_CS)

Moving to the right side of the table, we consider the crowdsourced difficulty as ground truth, and we present the results separately for i) the whole dataset, ii) the pairs of questions with Total Agreement between human annotators (TA), and iii) the pairs of questions with Total Agreement and Correctly Answered by each model (TA & CA).

The most crucial difference is that the best performing model, in this case, is ELMO_C. It means that leveraging the similarity between the provided document and the questions might be a good alternative to using the uncertainty of the QA models for comprehension MCQ. This is reasonable since the goal of comprehension questions is to find the answer to the question in the accompanying passage. However, this is not in agreement with the results obtained on PairRACE_HM. Also, we have to consider that PairRACE_CS is made of only 80 question pairs (37 when considering only the ones with total agreement) while PairRACE_HM contains 2M pairs; therefore the performance of ELMO_C is worth of further exploration. Moreover, BERT (E) and XLNet (E) clearly outperform ELMO_K and ELMO_QA, suggesting that indeed the uncertainty of accurate and calibrated QA models can be beneficial for QDE of knowledge questions (which are not provided with an accompanying passage that contains the answer).

Differently from the previous experiment, the performance of DistilBERT (E) here is clearly worse than the other ensembles (it is even worse than random), thus suggesting that – being a smaller model – they are not capable of modelling the questions as well as BERT (E) and XLNet (E).

Except for these two differences, the rest of the findings is fairly similar to PairRACE_HM, and the ensembles generally outperform the single instances of the same architecture. Crucially, the accuracy of all ensemble models (except the hybrid ones) is higher for pairs with the total agreement, thus supporting the claim that the uncertainty of QA models could really be used for unsupervised QDE from text. It is also interesting to remark that this is not always the case for the single instances of Transformers, which sometimes have worse accuracy on the pairs with total agreement. This, once again, suggests that calibrated ensembles are more suitable for unsupervised QDE from text.

Considering the pairs with the total agreement and containing questions correctly answered by the QA models (shown in column TA & CA^3), we can see that the correctness of the QA model does not seem to have a significant impact on the accuracy. However, there are only a few question pairs of this type; therefore we cannot perform any relevant observations.

7 Conclusions

The results of this research support the idea that it is possible to estimate the human-perceived difficulty of exam questions via the uncertainty scores produced by Question Answering (QA) neural networks. The advantage of the approach we propose is that it is possible to predict the relative difficulty of questions across different domains without needing any calibrated questions or logs of students’ answers. For training the QA model, it is sufficient to have access to i) the corpus of questions and (possibly) ii) the learning materials.

As a practical guideline, both the QA accuracy and the calibration seem to impact the accuracy of QDE. Therefore we believe that it is better to use calibrated models which are powerful enough to reach decent performance in the QA task (e.g. the BERT and XLNet ensembles used here).

Future work will focus on exploring whether improving the calibration of the QA models (e.g. increasing the number of models in the ensemble or using Bayesian neural networks) would lead to improved results in unsupervised QDE from text and will analyze the effects of combining raw softmax scores with different techniques. Weighing the

---

^3We do not show the accuracy unless there are at least 10 question pairs correctly answered.
scores by the accuracy of each intervening model is a way to explore whether hybrid ensembles can still improve the performance, but we leave the implementation of this approach for another study.

Another natural progression of this work is to leverage question-specific information, such as the reasoning type required to answer it, or the question format (e.g., cloze items vs interrogative items). Using the text of the possible choices might improve the accuracy, which makes it an interesting modification to explore in future studies. Further experimental investigations are also needed to use the proposed approach for creating a difficulty ranking of questions.

References


GeSERA: General-domain Summary Evaluation by Relevance Analysis

Jessica López Espejel∗†, Gaël de Chalendar∗, Jorge García Flores†, Thierry Charnois†, and Ivan Vladimir Meza Ruiz‡

* Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
gael.de-chalendar@cea.fr,
† CNRS-LIPN-Université Sorbonne Paris Nord, France
{jessicanayeli.lopezespejel, charnois}@lipn.univ-paris13.fr ; jgflores@lipn.fr
‡ IIMAS, UNAM, Mexico
ivanvladimir@turing.iimas.unam.mx

Abstract

We present GeSERA, an open-source improved version of SERA for evaluating automatic extractive and abstractive summaries from the general domain. SERA is based on a search engine that compares candidate and reference summaries (called queries) against an information retrieval document base (called index). SERA was originally designed for the biomedical domain only, where it showed a better correlation with manual methods than the widely used lexical-based ROUGE method. In this paper, we take out SERA from the biomedical domain to the general one by adapting its content-based method to successfully evaluate summaries from the general domain. First, we improve the query reformulation strategy with POS Tags analysis of general-domain corpora. Second, we replace the biomedical index used in SERA with two article collections from AQUAINT-2 and Wikipedia. We conduct experiments with TAC2008, TAC2009, and CNNDM datasets. Results show that, in most cases, GeSERA achieves higher correlations with manual evaluation methods than SERA, while it reduces its gap with ROUGE for general-domain summary evaluation. GeSERA even surpasses ROUGE in two cases of TAC2009. Finally, we conduct extensive experiments on the impact of the index size on summary evaluation with SERA and GeSERA. Code and Wikipedia dataset are publicly available.

1 Introduction

Automatic summary evaluation is a challenging task in Natural Language Processing (NLP). Evaluation is usually done by humans, but manual evaluation is subjective, costly and time expensive (Lin and Hovy, 2002). Automatic evaluation methods (Lin, 2004a; Torres-Moreno et al., 2010; Zhao et al., 2019; Zhang et al., 2020) are an alternative to save time for users who extract the most relevant content from the web using Automatic Text Summarization systems (ATS). There exist two types of evaluation approaches: (1) manual evaluation methods like Pyramid (Nenkova and Passonneau, 2004) and Responsiveness, where human intervention is mandatory, and (2) automatic evaluation methods, where human intervention can be needed as a ground-truth reference (Lin, 2004a; Cohan and Goharian, 2016) or not (Torres-Moreno et al., 2010; Cabrera-Diego and Torres-Moreno, 2018).

Summary Evaluation by Relevance Analysis (SERA) (Cohan and Goharian, 2016) is an automatic evaluation method that partially relies on human references to evaluate abstractive summaries from the biomedical domain. It was proposed as an alternative to ROUGE (Lin, 2004a), the widely used automatic metric, that is based on lexical overlaps between candidate and reference summaries. ROUGE is unfair to evaluate abstractive summaries where the ATS paraphrases the text instead of just copying-pasting chunks of it (Cohan and Goharian, 2016).

SERA is based on content relevance and is fairer to evaluate abstractive summaries because it attributes high scores to summaries that are lexically different but semantically related. However, it surpasses ROUGE on the biomedical domain only. In this paper, we modify SERA to make it usable for generic collections. We propose the following contributions:

1. Implement an open-source version of SERA from scratch.
2. Propose GeSERA (General-domain SERA), an improved version of SERA that is domain-independent.
3. Conduct extensive experiments with two large indexes (AQUAINT-2 (Graff, 2002) and Wikipedia) and three summarization datasets.
These datasets are well-suited for general-domain and news abstractive summary evaluation. GeSERA achieves competitive results compared to a range of state-of-the-art evaluation approaches on both abstractive and extractive summaries.

4. Make the code and our Wikipedia dataset publicly available to help future researchers.

5. Conduct extensive experiments on the impact of the index size on summary evaluation with SERA and GeSERA.

2 Proposed approach

2.1 Baseline: SERA

SERA (Cohan and Goharian, 2016) is based on a content relevance between a candidate summary and its corresponding human-written reference summaries using information retrieval. SERA compares these summaries (called queries) against a set of documents from the same domain (called index), and compares the overlap of retrieved results. SERA refines queries in three different manners: (1) Raw text - only stop words and numbers are removed, (2) Noun phrases (NP) - only noun phrases are kept, and (3) Keywords (KW) - only unigrams, bigrams, and trigrams are kept.

SERA scores are computed with Equation 1:

\[
\text{SERA} = \frac{1}{M} \sum_{i=1}^{M} \frac{|R_C \cap R_{G_i}|}{|R_C|}
\]

where: \(R_C\) is the list of retrieved documents for the candidate summary \(C\), \(R_{G_i}\) is the ranked list of retrieved documents for the reference summary \(G_i\), and \(M\) is the number of reference summaries.

Another variant of SERA is called SERA-DIS. It takes the order of retrieved documents into consideration (Equation 2).

\[
\text{SERA – DIS} = \frac{1}{M \cdot D_{\text{max}}} \sum_{i=1}^{M} \left( \sum_{j=1}^{R_C} \sum_{k=1}^{R_{G_i}} X_{j,k} \right)
\]

\[
X_{j,k} = \begin{cases} 
1 & \text{if } R_C(j) = R_{G_i}(k) \\
0 & \text{otherwise}
\end{cases}
\]

where: \(R_C(j)\) is the \(j^{th}\) result in the ranked list \(R_C\), and \(D_{\text{max}}\) is the maximum achievable score used for normalization. In both SERA variants, retrieved results are truncated at 5 and 10 documents (hence the notations SERA-5 and SERA-10 in Section 3). Cohan and Goharian (2016) used articles from PubMed as an index, and summaries from TAC 2014 as queries.

The intuition behind SERA is that a summary context is represented by its most related articles. Thus, two summaries related to the same documents are semantically related, even if they are lexically different. Consequently, SERA is fairer to evaluate abstractive summaries contrarily to the lexical-based ROUGE. However, SERA suffers from a series of limitations: (1) the code is not open-source, (2) no information was provided concerning the subset of PubMed used as an index, and (3) PubMed is specialized in the biomedical domain only. The first two drawbacks make SERA unusable by the community, while the third restricts its usage to the biomedical domain.

2.2 GeSERA: General-domain SERA

We build on SERA merits and limitations to propose GeSERA, an open-source version of SERA that evaluates summaries from the general domain. Novelties of GeSERA are the index pool and query reformulation adapted to the evaluation of summaries from the general domain.

Index documents pool - Differently from SERA, GeSERA enables general-domain summary evaluation. It is thus necessary to replace the biomedical index used by Cohan and Goharian (2016) with a set of documents related to the general domain. We build many indexes using a variant number of articles from Wikipedia and AQUAINT-2. See Subsection 3.1 for more details.

Query reformulation (QR) - It improves retrieval process by removing unnecessary terms from the text. Therefore, we propose a different approach to refine queries in GeSERA that is better suited for general domain summaries.

According to Kieuvonggnam et al. (2020), nouns in generated summaries represent more accurately the information conveyed by the original abstracts than other POS tags. This study was conducted on Covid-19 medical texts and can explain why SERA achieved a higher correlation than ROUGE for the TAC 2014 biomedical dataset. We led an analysis that consists of analyzing Part-Of-Speech (POS) tags distribution

\[
\text{https://tac.nist.gov/}
\]

\[
\text{https://github.com/}
\]

\[
\text{JessicaLopezEspejel/GeSERA/}
\]

\[
\text{http://www.ncbi.nlm.nih.gov/pmc/}
\]
for PubMed (biomedical dataset built by Cohan et al. (2018)), AQUAINT-2 (news corpus), and Wikipedia (general domain encyclopedia). Figure 1 shows bar plots for percentages of nouns, verbs, adjectives (Adj.), prepositions (Prep.), and the total percentage of other tags (Others).

Figure 1: POS Tags distribution percentages for Wikipedia, AQUAINT-2, and PubMed datasets

Our analysis of three datasets which describe different domains confirms the observation of Kieuvongngam et al. (2020) for PubMed. However, it shows that the percentages of verbs and adjectives are higher in AQUAINT-2 and Wikipedia than in PubMed. Equally important, there is a remarkable absence of prepositions in Wikipedia and AQUAINT-2. Based on our analysis, we propose to reformulate queries in GeSERA by only keeping tokens tagged with nouns, verbs, and adjectives, the three most frequent tags in the news and general domain corpora.

Search engine - A semantic-based retrieval approach is crucial when handling abstractive summaries. In order to compare the queries against the index, a search engine is needed for information retrieval and scoring. We use the Whoosh4 search engine with the BM25F (Best Match 25 Model with Extension to Multiple Weighted Fields) ranking function (Zaragoza et al., 2004). This model is widely used for semantic search (Pérez-Agüera et al., 2010; Robertson and Zaragoza, 2009). It consists in weighting terms according to their field importance, combining them, and using the resulting pseudo-frequencies for ranking.

3 Experiments

SERA was developed in the context of scientific biomedical article summarization with the idea that its semantic specificity is particularly useful for this domain. We hypothesize that if we reformulate queries properly and change the index pool, SERA can assess summaries from other domains for both abstractive and extractive summarization. This hypothesis is based on the fact that SERA considers terms that are not lexically equivalent but are semantically related. We conduct extensive experiments on SERA and GeSERA to test our hypothesis.

3.1 Index datasets

The index is a key component of GeSERA approach insofar as it should describe the same domain as the queries. The number of documents in the index is also decisive as we will show in Subsection 4.1. We describe briefly query and index datasets hereafter and provide more information in Appendix B.

AQUAINT-2 (Graff, 2002) is a news corpus built from various sources. We vary the size of the index to include \( I = \{10000, 15000, 30000, 60000, 89760, 179520, 825148\} \) randomly-selected documents.

Wikipedia is a free encyclopedia that contains various information from the general domain. We vary the size of the index to include \( I = \{10000, 15000, 30000, 1778742\} \) randomly-selected documents.

3.2 Query datasets

Candidate and reference summaries from the news datasets TAC2008, TAC2009, and the CNN Daily Mail (CNNDM) version published by Bhandari et al. (2020) are used as queries.

TAC2008 (and TAC2009) are subsets of AQUAINT-2. They contain 5568 (4840) candidate summaries proposed by 58 (55) participants, and 384 (352) reference summaries, respectively. These datasets are designed for multi-document extractive summarization (one summary is shared by a set of documents).

CNN Daily Mail (Bhandari et al., 2020) is a news dataset, and is of great interest to us because it has candidate summaries obtained from both extractive and abstractive systems. It consists of 100 reference summaries, having each 25 candidate summaries generated by 11 extractive and 14 abstractive systems. It is designed for monodocument abstractive and extractive summarization (one summary for each document).
We compare GeSERA with some of the most influential evaluation metrics from the literature:

- **ROUGE** and **SERA** - two automatic evaluation approaches that rely on human intervention. ROUGE has many variants, but we only report the most popular ones: ROUGE-N ($N = \{1, 2\}$ is the n-gram size) and ROUGE-L (Longest Common Subsequence). For each variant, we report the F-score, Recall and Precision.

- **MoverScore** (Zhao et al., 2019) and **BERTScore** (Zhang et al., 2020) - two automatic evaluation approaches based on BERT.

- **JS-2** (Lin et al., 2006) - Jensen-Shannon divergence between bigram’s distribution of the candidate and reference summaries.

- **SummTriver** (ST) (Cabrera-Diego and Torres-Moreno, 2018) - based on trivergence, i.e. a composition ($T_c$) or multiplication ($T_m$) of Kullback-Leibler (KL), Jensen-Shannon (JS), and smoothed Jensen-Shannon (sJS) divergences. It does need a human reference.

- **FRESA** (Torres-Moreno et al., 2010) - an automatic evaluation method that do not rely on human intervention. It is based on Jensen-Shannon divergence and has four variants: unigrams (FRESA-1), bigrams (FRESA-2), trigrams (FRESA-3), and skip-grams (FRESA-4).

More information is in Section 5. Implementation details are in Appendix C.

### 4 Results and discussion

In Subsection 4.1, we vary the index size in SERA and GeSERA and study the variation of their performance on TAC datasets by averaging the score of all four manual annotators. Once we determine the best index size, we present in Subsection 4.2 the correlations using the best index size of each method.

#### 4.1 Impact of the index size on the performance of SERA and GeSERA

We first present the impact of the indexes size built from Wikipedia on SERA and GeSERA, then those built from AQUAINT-2.

**Wikipedia Index** - Figures 2-b and 2-d show Pearson correlations of SERA and GeSERA with Pyramid when indexing different values of $I$ from the Wikipedia dataset, and when using TAC2008 and TAC2009 as query datasets, respectively. Figures show that the best score (0.902) is obtained with SERA-DIS-NP-10 using $I = 30,000$ for TAC2008 while the best one (0.957) for TAC2009 is obtained with GeSERA-DIS-10 with $I = 10,000$. We thus use in Subsection 4.2 an index size $I = 30,000$ and $I = 10,000$ for TAC2008 and TAC2009, respectively. Surprisingly, the worst scores are obtained with $I = 1,778,742$, the largest and more diversified index size corresponding to all documents from our Wikipedia corpus.

**AQUAINT-2 Index** - Figures 2-a and 2-c show the Pearson correlation coefficients of SERA and GeSERA with Pyramid when indexing different values of $I$ from the AQUAINT-2 dataset, and when using TAC2008 and TAC2009 as query datasets. Similarly to Wikipedia, figures show that overall, the best results are obtained with small index sizes. The best score (0.928) was obtained with GeSERA-5 using $I = 15,000$ for TAC2008, while the best one (0.947) for TAC2009 is obtained with both SERA-DIS-NP-10 and GeSERA-DIS-5 using $I = 179,520$. Note that results with $I = 10,000$ are comparable with those obtained with $I = 179,520$ for TAC2009. We thus use in Subsection 4.2 an index size $I = 15,000$ and $I = 179,520$ for TAC2008 and TAC2009, respectively. Once again, the lowest results are obtained with the full AQUAINT-2 corpus corresponding to $I = 825,148$. 

---

864
Figure 2: Pearson correlation coefficients using TAC2008 and TAC2009 as queries, and AQUAINT-2 and Wikipedia as indexes. Best viewed in color.

4.2 Comparison of GeSERA with the SoTA
We use the best index sizes from Subsection 4.1 to report the detailed results with the best variants of each method from Subsection 3.3. More results are reported in Appendix D.

4.2.1 TAC2008 query dataset
Table 1-up shows correlations of the best variants of SERA, GeSERA, ROUGE, SummTriver (ST) and FRESA with two manual evaluation approaches: Pyramid and Responsiveness. Note that for SERA and GeSERA, we fix the query dataset TAC2008, while we vary the index between AQUAINT-2 and Wikipedia.

AQUAINT-2 Index - Results show that GeSERA-5 achieves the best scores among all variants of SERA and GeSERA for both Pyramid and Responsiveness. SERA-5 is the best variant of SERA for Pyramid with Pearson and Spearman, while SERA-NP-10 provides better results for Pyramid with Kendall, and Responsiveness with all correlation measures. GeSERA-5 surpasses the two best variants of SERA by 0.015 (Pearson), 0.016 (Spearman), and 0.021 (Kendall) points for Pyramid, and by 0.02, 0.027, and 0.031 points for Responsiveness. The gains are higher for Responsiveness, and for Kendall correlations.

Wikipedia Index - Results show that GeSERA-10 is the best variant of GeSERA for Pyramid with all correlation metrics used. GeSERA-DIS-10 gets the best scores for Responsiveness with Pearson and Kendall. Alternatively, the best SERA variant is SERA-DIS-NP-10 for Pyramid with all correlation measures, and with Responsiveness for Spearman and Kendall measures. Interestingly, when using queries from TAC2008 with Wikipedia, GeSERA does not outperform SERA neither for Pyramid, not for Responsiveness.

<table>
<thead>
<tr>
<th>TAC2008</th>
<th>Pyramid</th>
<th>Responsiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pearson</td>
<td>Spearman</td>
<td>Kendall</td>
</tr>
<tr>
<td>SERA-ST</td>
<td>0.896</td>
<td>0.831</td>
</tr>
<tr>
<td>SERA-NP</td>
<td>0.885</td>
<td>0.828</td>
</tr>
<tr>
<td>SERA-DIS</td>
<td>0.941</td>
<td>0.836</td>
</tr>
<tr>
<td>SERA-DIS-NP</td>
<td>0.947</td>
<td>0.825</td>
</tr>
<tr>
<td>GeSERA-ST</td>
<td>0.908</td>
<td>0.835</td>
</tr>
<tr>
<td>GeSERA-DIS</td>
<td>0.947</td>
<td>0.836</td>
</tr>
</tbody>
</table>

Wikipedia Index | Pearson | Spearman | Kendall | Pearson | Spearman | Kendall |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SERA-ST</td>
<td>0.894</td>
<td>0.832</td>
<td>0.871</td>
<td>0.816</td>
<td>0.892</td>
<td>0.590</td>
</tr>
<tr>
<td>SERA-NP</td>
<td>0.885</td>
<td>0.828</td>
<td>0.862</td>
<td>0.806</td>
<td>0.762</td>
<td>0.529</td>
</tr>
<tr>
<td>SERA-DIS</td>
<td>0.941</td>
<td>0.836</td>
<td>0.871</td>
<td>0.806</td>
<td>0.873</td>
<td>0.521</td>
</tr>
<tr>
<td>SERA-DIS-NP</td>
<td>0.947</td>
<td>0.825</td>
<td>0.865</td>
<td>0.806</td>
<td>0.883</td>
<td>0.518</td>
</tr>
<tr>
<td>GeSERA-ST</td>
<td>0.908</td>
<td>0.835</td>
<td>0.878</td>
<td>0.834</td>
<td>0.897</td>
<td>0.530</td>
</tr>
<tr>
<td>GeSERA-DIS</td>
<td>0.947</td>
<td>0.836</td>
<td>0.868</td>
<td>0.831</td>
<td>0.884</td>
<td>0.525</td>
</tr>
</tbody>
</table>

Table 1: Correlations on TAC2008 and TAC2009 datasets, in terms of Pearson, Spearman and Kendall, of automatic evaluation methods with Pyramid and Responsiveness. The best first (red), second (blue) and third (black) score of each column are in bold.
While ROUGE-2-R and ROUGE-3-F provide the best results for all correlation measures on TAC2008, GeSERA and SERA largely outperform the scores of SummTriver and FRESA with both Pyramid and Responsiveness. In the case of GeSERA-5, it achieves higher correlations than ST-JS-Tm, the best variant of SummTriver, by 0.039, 0.097, and 0.117 for Pyramid, and 0.049, 0.053, and 0.055 for Responsiveness. Finally, FRESA baseline achieves the lowest correlation scores in all configurations. The performance of SummTriver and FRESA is not surprising insofar as they do not rely on any human reference.

4.2.2 TAC2009 query dataset

Table 1 shows correlation coefficients of SERA, GeSERA, ROUGE, SummTriver and FRESA with two manual evaluation approaches: Pyramid and Responsiveness. Once again, we fix the query dataset TAC2009, while we vary the index between AQUAINT-2 and Wikipedia.

AQUAINT-2 Index - ROUGE provides the highest scores against SERA and GeSERA when we index documents from AQUAINT-2. Importantly, GeSERA-DIS-5 and GeSERA-5 achieve higher correlations than SERA with Pyramid and Responsiveness, respectively. Note that the scores of SERA vary more between its variants, while results of GeSERA are more stable and the best ones are obtained with only two of its variants. This finding highlights the robustness of our approach against variations of configurations.

Wikipedia Index - SERA and GeSERA outperform ROUGE against the Pyramid manual method in terms of Pearson correlation when indexing documents from Wikipedia. The best correlations are obtained by GeSERA-DIS-10 and SERA-10 against Pyramid and Responsiveness, respectively. Interestingly, for TAC2009, GeSERA-DIS-10 achieves better Pearson correlation than ROUGE with Pyramid, and GeSERA-10 with Responsiveness. This finding proves the effectiveness of GeSERA to evaluate summaries from the general domain. Equally, GeSERA reduces the gap between SERA and ROUGE in most of other cases. SummTriver achieves reasonably good results in Table 1 even without the use of any human reference. This baseline is useful when human summaries are costly or hard to find. However, when such references are available, SummTriver does not take advantage of them, leading its correlation to be low compared to human-based evaluation approaches such as ROUGE and SERA.

FRESA shows the lowest scores among evaluation approaches tested here. It drops approximately from 0.1 to 0.3 point compared to the lowest results obtained by SERA. This is mainly because FRESA is based only on the divergence between the evaluated summary and its source documents, without including any comparison with summaries generated by other participants, as SummTriver does. Thus, FRESA is barely correlated with manual evaluation in many cases where the correlation gets so close to zero (for instance, FRESA-2 with TAC2009 using Kendall correlation). Note that SummTriver and FRESA have mostly negative correlations because they are based on a divergence measure which increases when the summary’s quality is low and decreases when its quality is high.

4.2.3 CNNDM query dataset

Based on results obtained in Subsection 4.1 regarding the effectiveness of SERA and GeSERA with small index sizes, we decided to index \( I = 10000 \) documents from Wikipedia to run the two methods on CNNDM.

<table>
<thead>
<tr>
<th>CNNDM</th>
<th>Pearson</th>
<th>Spearman</th>
<th>Kendall</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROUGE-1-R</td>
<td>0.914</td>
<td>0.922</td>
<td>0.773</td>
</tr>
<tr>
<td>ROUGE-2-R</td>
<td>0.962</td>
<td>0.958</td>
<td>0.860</td>
</tr>
<tr>
<td>ROUGE-L-F</td>
<td>0.526</td>
<td>0.368</td>
<td>0.278</td>
</tr>
<tr>
<td>BERTScore-1-P</td>
<td>-0.021</td>
<td>0.093</td>
<td>0.064</td>
</tr>
<tr>
<td>BERTScore-1-R</td>
<td>0.768</td>
<td>0.738</td>
<td>0.552</td>
</tr>
<tr>
<td>MoverScore</td>
<td>0.443</td>
<td>0.367</td>
<td>0.284</td>
</tr>
<tr>
<td>JS-2</td>
<td>0.780</td>
<td>0.665</td>
<td>0.512</td>
</tr>
<tr>
<td>SERA-10</td>
<td>0.885</td>
<td>0.782</td>
<td>0.621</td>
</tr>
<tr>
<td>SERA-KW-10</td>
<td>0.864</td>
<td>0.782</td>
<td>0.605</td>
</tr>
<tr>
<td>SERA-DIS-10</td>
<td>0.827</td>
<td>0.781</td>
<td>0.391</td>
</tr>
<tr>
<td>SERA-DIS-NP-5</td>
<td>0.599</td>
<td>0.554</td>
<td>0.391</td>
</tr>
<tr>
<td>GeSERA-5</td>
<td>0.623</td>
<td>0.527</td>
<td>0.387</td>
</tr>
<tr>
<td>GeSERA-10</td>
<td>0.880</td>
<td>0.872</td>
<td>0.719</td>
</tr>
<tr>
<td>GeSERA-DIS-10</td>
<td>0.817</td>
<td>0.788</td>
<td>0.605</td>
</tr>
</tbody>
</table>

Table 2: Correlation coefficients on CNNDM, in terms of Pearson, Spearman and Kendall, of multiple automatic evaluation methods with LitePyramid.

Table 2 shows that the highest correlations of ROUGE are obtained with ROUGE-2-R, followed by ROUGE-1-R. Globally, the highest correlations in ROUGE are obtained with the recall metric (ROUGE-R), followed by ROUGE-F, and finally by ROUGE-P. The following highest correlations are obtained with GeSERA-10. Once again, GeSERA surpasses all the SERA variants, and the state-of-the-art methods presented in this table.

866
Although SERA-KW-10 has the best score in terms of Pearson and Kendall, all SERA variants present very similar scores. Behind the SERA method, BERTScore and JS-2 measures present very similar scores. Meanwhile, MoverScore shows the lowest correlations. Results show the effectiveness of GeSERA to evaluate extractive and abstractive summaries since CNNDM contains both approaches.

5 Related work

Evaluation methods are fundamental techniques to assess if summaries generated by an automatic system capture the original document’s idea. Different evaluation methods have been developed in the last decade for the evaluation of automatically-generated summaries. There exist two types of evaluation methods: (1) manual evaluation methods and automatic evaluation methods. The first group of methods requires human intervention as ground-truth references. Pyramid (Nenkova and Passonneau, 2004) and Responsiveness are the most popular such methods. The second group of methods is divided itself into two subsets: (1) methods that need human intervention like ROUGE (Lin, 2004a) and SERA (Cohan and Goharian, 2016), and (2) methods that do not need any human reference like SummTriver (Cabrera-Diego and Torres-Moreno, 2018) and FRESA (Torres-Moreno et al., 2010).

The most popular automatic metric used by the community is ROUGE (Lin, 2004a). It needs reference summaries, and is based on their lexical overlaps with candidate summaries. That is why it is more useful to evaluate extractive summaries where chunks of the text are copied and pasted to form the summary. However, in the case of abstractive summaries where the ATS paraphrases the text with possibly new vocabulary, the ROUGE metric becomes unfair. To overcome this issue, researchers have been proposing in the last few years other automatic metrics to fairly evaluate both extractive and abstractive summaries.

The first type of automatic evaluation methods relies partially on human judgment as ROUGE does. The simplest method is Jensen-Shannon (JS-2) (Lin et al., 2006). It is based on the divergence between bi-gram’s distribution of candidate and reference summaries. More sophisticated systems include MoverScore (Zhao et al., 2019) that is based on fine-tuning the BERT model and combining contextualized representations with Earth Mover Distance (EMD) from (Rubner et al., 2000). BERTScore (Zhang et al., 2020) also is based on BERT model. Unlike ROUGE (Lin, 2004b), BERTScore makes use of contextual embeddings that are effective for paraphrase detection. Similarly to BERTScore, Semantic Similarity for Abstractive Summarization (SSAS) (Vadapalli et al., 2017) is based on semantic matching between candidate and reference summaries. The second type of automatic evaluation methods does not need any human intervention. For instance, FRamework for Evaluating Summaries Automatically (FRESA) (Torres-Moreno et al., 2010) is based on divergences among probability distributions between the summary to evaluate and its source document. Another well-known metric is SummTriver (Cabrera-Diego and Torres-Moreno, 2018). It is based on Trivergences between the summary to evaluate, its source document(s), and a set of summaries related to the same source document(s) but generated with other ATS systems.

6 Conclusion and perspectives

We introduced GeSERA, an open-source system for general-domain summary evaluation. We redefine query reformulation of SERA based on POS Tags analysis of datasets from different domains, and change the biomedical index with documents from AQUAINT-2 and Wikipedia. GeSERA achieves competitive results compared to state-of-the-art approaches. Overall, GeSERA surpasses SERA and reduces its gap with ROUGE, and in two cases, it even surpasses ROUGE, the lexical-based method. Unsurprisingly, the comparison with evaluation methods that do not rely on human references reveals a large gap in favor of GeSERA since it relies on human references while the others do not. Finally, extensive experiments show that the index size has a considerable effect on the performance of SERA and GeSERA that tend to perform better with small-size indexes. Our code is publicly available to facilitate reproducibility.

We will: (1) explore other variants of the search engine to know its impact on GeSERA, (2) propose a new version of GeSERA that does not rely on human intervention by exploiting information from the source text, (3) apply prepossessing on the index and search for other solutions to improve query reformulation, and (4) explore larger query datasets such as Multi-News (Fabbri et al., 2019).
References


A Introduction

In this appendix, we provide: (1) more details about evaluation datasets (Sec. B), (2) implementation details (Sec. C), and (3) detailed results of all tested approaches (Sec. D).

B Datasets details

- **AQUAINT-2** is a news corpus built from New York Times, Associated Press, and Xinhua News Agency. Indexes built from this corpus are balanced, except for the largest one ($I = 825, 148$), that contains all documents.

- **Wikipedia** is a free online encyclopedia from the general domain. The largest index ($I = 1, 778, 742$) contains all available documents.

- **TAC2008** (TAC2009) contains two sets. Each set contains 48 (44) topics. Each topic includes 10 documents and 4 reference summaries. Candidate summaries are proposed by 58 (55) participants, where each one provides a candidate summary per topic. In total, there are 960 (880) documents, 5568 (4840) candidate summaries, and 384 (352) reference summaries.

C Implementation details

SERA and GeSERA were implemented in Python. For information retrieval, we used the Okapi BM25F ranking function from Whoosh, a flexible and pure python search engine framework.

<table>
<thead>
<tr>
<th></th>
<th>Pearson</th>
<th>Spearman</th>
<th>Kendall</th>
</tr>
</thead>
<tbody>
<tr>
<td>JS-2</td>
<td>0.780</td>
<td>0.665</td>
<td>0.512</td>
</tr>
<tr>
<td>SERA-5</td>
<td>0.773</td>
<td>0.710</td>
<td>0.508</td>
</tr>
<tr>
<td>SERA-NP-5</td>
<td>0.690</td>
<td>0.639</td>
<td>0.452</td>
</tr>
<tr>
<td>SERA-NP-10</td>
<td>0.743</td>
<td>0.705</td>
<td>0.502</td>
</tr>
<tr>
<td>SERA-KW-5</td>
<td>0.748</td>
<td>0.714</td>
<td>0.509</td>
</tr>
<tr>
<td>SERA-DIS-5</td>
<td>0.786</td>
<td>0.688</td>
<td>0.472</td>
</tr>
<tr>
<td>SERA-DIS-NP-5</td>
<td>0.671</td>
<td>0.568</td>
<td>0.393</td>
</tr>
<tr>
<td>SERA-DIS-KW-5</td>
<td>0.758</td>
<td>0.657</td>
<td>0.465</td>
</tr>
<tr>
<td>SERA-DIS-KW-10</td>
<td>0.828</td>
<td>0.752</td>
<td>0.565</td>
</tr>
<tr>
<td>GeSERA-DIS-5</td>
<td>0.566</td>
<td>0.469</td>
<td>0.315</td>
</tr>
</tbody>
</table>

Table 3: Correlations of CNNDM dataset, in terms of Pearson, Spearman and Kendall, of multiple automatic evaluation methods with LitePyramid.

<table>
<thead>
<tr>
<th></th>
<th>Pearson</th>
<th>Spearman</th>
<th>Kendall</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST-JS-7P</td>
<td>0.882</td>
<td>-0.302</td>
<td>0.517</td>
</tr>
<tr>
<td>ST-KL-7P</td>
<td>-0.694</td>
<td>-0.700</td>
<td>-0.510</td>
</tr>
<tr>
<td>ST-JS-5P</td>
<td>0.858</td>
<td>-0.805</td>
<td>0.614</td>
</tr>
<tr>
<td>ST-JS-7T</td>
<td>-0.857</td>
<td>-0.805</td>
<td>-0.612</td>
</tr>
<tr>
<td>ST-KL-7T</td>
<td>-0.216</td>
<td>-0.168</td>
<td>-0.121</td>
</tr>
</tbody>
</table>

Table 4: Correlations on TAC2008 and TAC2009.
We used the authors’ public implementations to run ROUGE, SummTriver, and FRESA. The latter was basically designed for mono-document evaluation. Thus, we concatenated all the articles of the same topic to be able to run it on TAC.

To compute the correlations with LitePyramid of ROUGE, BERTScore, MoverScore, and JS-2 on the CNN Daily Mail dataset, we used the scores provided by Bhandari et al., 2020 in their GitHub repository\(^5\). Based on experiments on the TAC datasets in Subsection 4.3, we use an index size of \(I = 10,000\) in SERA and GeSERA.

For the sake of comparability, scores are averaged for each participant before computing the correlations with manual methods.

D More results

Table 3 and Table 4 provide more results on CN-NDM, TAC2008 and TAC2009 datasets. Both tables present variants of the evaluation metrics that we did not report in the main paper.

\(^5\)https://github.com/neulab/REALSumm
On the Interaction between Annotation Quality and Classifier Performance in Abusive Language Detection

Holly Lopez Long, Alexandra O’Neil, Sandra Kübler
Indiana University
Bloomington, IN, USA
{hdlopezl,aconeil,skuebler}@iu.edu

Abstract
Abusive language detection has become an important tool for the cultivation of safe online platforms. We investigate the interaction of annotation quality and classifier performance. We use a new, fine-grained annotation scheme that allows us to distinguish between abusive language and colloquial uses of profanity that are not meant to harm. Our results show a tendency of crowd workers to overuse the abusive class, which creates an unrealistic class balance and affects classification accuracy. We also investigate different methods of distinguishing between explicit and implicit abuse and show lexicon-based approaches either over- or under-estimate the proportion of explicit abuse in data sets.

1 Introduction
In recent years, annotation quality has come under closer scrutiny, especially for subjective classification tasks that rely on human judgement. Investigations of unintended bias in abusive language data sets have demonstrated that they are susceptible to sampling and annotation bias (Wiegand et al., 2019; Sap et al., 2019). Although this work provides some guidance for reducing the effects of unintended bias from sampling, it does not provide a clear path forward for mitigating annotation bias. As we need effective ways to curb online hate, we definitely need reliable data sets with high-quality annotations for abusive language detection.

In this paper, we compare annotations by untrained crowd workers with annotations by experts. Our examination demonstrates that labeling differences between crowd workers and experts change the class distribution in the data set and affect classifier performance. We also compare methods for determining explicit and implicit abuse in the data set, and how this affects the interpretation of machine learning experiments. Our paper is structured as follows: Sec. 2 explains our research questions, sec. 3 describes prior work on data sets, annotation procedures and quality, and classification schemes for abusive language, sec. 4 describes our data sets and methodology, sec. 5 discusses our insights into the interaction of annotation quality and classifier performance, and sec. 6 investigates the interaction of explicit and implicit abuse and the interpretation of results. We conclude in sec. 7.

2 Research Questions
We started our investigation by reviewing a random sample of 1000 posts from the Kaggle competition Jigsaw Unintended Bias in Toxicity Classification. Our initial inspection showed that many posts that were considered abusive by crowd annotators (wrt. the final classification used by the competition) were open to interpretation. The examples below show typical “abusive” posts where different interpretations are possible.

1. I do love Yataimura Maru’s ramen? It is a perfect food for Portland’s long winter. And PDX does kick a little ass.
2. Sorry to have to do this, but just to see if profanity filtering is enabled: fuck.
3. Took this as an opportunity to check back in on The Yard and the floorplans are finally up and they are ATROCIOUS.

The first example is a positive review of a ramen restaurant in Portland that also contains profanity. The second example also uses profanity, but is doing so as a part of a meta-comment about the filters used by the platform. Finally, the last post is a criticism of an apartment complex. Although an insult is used, it is directed toward an object and not
individuals. These examples show two common tendencies by crowd workers: interpreting profanity as abusive without considering the context and not distinguishing between insults and criticism directed at people and objects. These observations led us to our first research question:

**RQ1:** How does annotation quality affect characteristics of the data set and subsequently the performance of machine learning approaches? More specifically, we address the following questions:

**RQ1.1:** How does annotation quality affect the distribution of abusive and non-abusive posts?

**RQ1.2:** Based on a detailed annotation scheme that distinguishes varieties of non-abuse, how do crowdsourced and expert annotations differ?

**RQ1.3:** How do annotations by crowd workers and experts influence classification results?

We then turn to the issue of explicit vs. implicit abuse. It is generally accepted that explicit abuse is easier to detect automatically (Wiegand et al., 2019). However, the method that we use to determine whether a post is explicitly or implicitly abusive will result in different splits of the data, and different results of how well a classifier performs on either class. Explicit abuse is often identified via lexicons of abusive expressions. Automatically created lexicons, such as the one by Wiegand et al. (2018), have good coverage, but may overestimate the abusiveness of terms while manually curated lists, such as the one by Razo and Kübler (2020), are more reliable in their selection of abusive terms but may lack coverage. This leads to our second question:

**RQ2:** How does the method of identifying explicit abuse influence the distribution in the data set and subsequently the interpretation of the classifier’s performance?

### 3 Related Work

#### Data Sets and Their Development

There is an abundance of data sets available for abusive language detection, which represent a variety of approaches for annotating abusive content. While many data sets have relied on large pools of crowd sourced annotators (Zampieri et al., 2019), others have used experts (Waseem and Hovy, 2016). Crowdsourcing annotations is often an attractive option for developing abusive language data sets, since the process often requires a considerable amount of time and labor. The two largest data sets were created for Kaggle competitions: 1) The *Toxic Comment Classification Challenge*[^1], which contains 312,737 posts from Wikipedia Talkpages, and 2) the *Jigsaw Unintended Bias in Toxicity Classification* (see Section 4.1) with posts from the platform Civil Comments. Both data sets were annotated by crowd workers. However, using crowd workers can contribute to diminished annotation quality (Hsueh et al., 2009). Waseem (2016) found that amateur annotators were more likely to label a post as hate speech and expert annotations improved machine learning performance. Sap et al. (2019) showed amateur annotators more often labeled African American English posts as abusive, but that priming the amateur annotators for dialect and race reduced annotation bias.

#### Annotation Schemes

One way to maintain annotation quality is to create clear annotation guidelines with rich taxonomies (Vidgen and Derczynski, 2020). Many scholars have developed annotation schema and guidelines that describe different types of abuse in order to better characterize abusive content. Founta et al. (2018) evaluated 7 abuse categories (e.g., offensive, abusive, hateful, aggressive, cyberbullying, spam, and normal), which were then merged into four (e.g., abusive, hateful, spam, and normal) when they found overlap between categories. Zampieri et al. (2019) created a 3-tier scheme in which annotators decided whether a post was abusive, targeted, and whether the target was an individual, a group, or other. Davidson et al. (2017) distinguished hateful content from the casual use of profanity by creating three categories: hateful, offensive (but not hateful), and neither. Current methods overwhelmingly focus on the labeling of abusive posts, often at the expense of accuracy on non-abusive posts.

#### Investigating Unintended Bias

Recent work on abusive language detection has looked at sampling bias in the data. Sampling methods are required to increase the amount of abusive posts in data sets. However, the specific sampling methods used have been shown to create bias. Wiegand et al. (2019) document this bias. Razo and Kübler (2020) build on their work and find that the source of the text (Twitter, Wikepedia, etc.) has more influence on the bias of the data set than the sampling method.

4 Methodology

4.1 Data
We use subsets of the data set from the Kaggle competition Jigsaw Unintended Bias in Toxicity Classification with posts from the platform Civil Comments. For the Jigsaw challenge, each comment was annotated by several crowd workers and a mean annotation score of $\geq 0.5$ (range: 0.0–1.0) was considered abusive. From this data set, we use two sampling subsets by Razo and Kübler (2020): the first subset of the random boosted sampling sets (a random sample) and the first of the biased topic sampling sets (increasing the number of abusive posts by searching for controversial topics that tend to attract abuse)\(^3\).

4.2 Classifier Settings
For the machine learning experiments below, we follow similar procedures as Razo and Kübler (2020). As Razo and Kübler, we use SVMs, more specifically, the SVC class of Scikit-learn (Pedregosa et al., 2011) with the RBF kernel, the same parameter settings (e.g., $C=1000$, $\text{gamma}=0.001$), and word 1-3 grams for features. We also perform 5-fold cross validation. Unlike Razo and Kübler, we do not remove punctuation.

5 Investigating Annotation Quality
As discussed in section 2, a cursory inspection of the data sets showed that there was a considerable amount of posts that were annotated as abusive (based on Jigsaw’s definition of the challenge), which the expert annotators found questionable. This does not only mean that the classifier learns a model that is disposed towards classifying too many posts as abusive, it also raises the question of whether a more consistent annotation would improve classification results or make the task more difficult to learn (since profanity would have to be disambiguated between abusive and colloquial use). For this reason, we decided to re-annotate the abusive portions of the two data sets. We first present the new annotation scheme in section 5.1, then we describe the resulting changes to the data set in section 5.2 and on the classifier in section 5.3.

5.1 New Annotation Scheme
The new annotation scheme includes 8 categories: explicit, implicit, self-abusive, irony, colloquial, meta, argumentative, and non-abusive. The categories of explicit, implicit, and self-abuse are considered to be abusive, all other categories are non-abusive. These categories were developed using a grounded theory approach (Glaser and Strauss, 1967; Corbin and Strauss, 2014), where the researchers open-coded a small set of instances originally labeled abusive and then consolidated categories and refined definitions. During the open-coding stages, researchers focused on characteristics that non-experts erroneously consider abusive.

We created categories to capture the challenging nuances of language, such as discussions about abuse (meta) and argumentative statements that may be antagonistic to a particular idea or policy, but not abusive toward individuals. While many existing schemes focus on distinctions between different varieties of abuse (Founta et al., 2018), our scheme \(^4\) focuses on non-abusive instances including profanity, etc. The categories with examples are shown in Table 1. The last category is for posts that were originally labeled as non-abusive.

The category explicit describes posts that use insults, threats, ethnic/religious slurs, and/or ad hominem attacks. This included instances of cyberbullying (e.g., attacking people’s appearance/body shape) and other forms of overt hate based on attributes of their identity, such as religion, ethnicity, sexuality, disability, or socioeconomic class.

The category implicit is used to indicate that a post degrades individuals or groups of people by alluding to stereotypes or other insulting speech through indirect methods. These posts include the same stereotypes apparent in explicit abuse, but instead of being directly expressed, the abuse is implied in the post.

Self-abuse is used to label posts in which people direct the abuse against themselves. While it could be argued that people should have the right to abuse themselves, we group this category with the other abusive categories because certain types of self-abuse may result in a diminished sense of self-worth (cf. e.g., negative self-talk).

As posts can often belong to several categories, especially since posts are often longer than Twitter posts, we label each post with all applicable labels. However, for all machine learning experiments, we reduced the annotation automatically to a single label per post, either abusive or non-abusive, to keep

\(^3\)Available at https://github.com/danterazo/abusive-language-detection/

\(^4\)https://github.com/hlopezlong/Annotation_Quality/blob/main/AnnotationGuidelines.txt
### Category Examples

<table>
<thead>
<tr>
<th>Category</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit</td>
<td><em>Liberals are just bone stupid. There can be no other rational explanation for their bias and ignorance.</em></td>
</tr>
<tr>
<td>implicit</td>
<td><em>Trump loves his uneducated voters. It sounds like you know a few yourself.</em></td>
</tr>
<tr>
<td>self-abuse</td>
<td><em>I not only missed the point, I missed the headline. I screwed up. I attempted to delete my idiotic comment several times but it keeps reappearing. Stupid is as stupid does and I sure did stupid (to slightly misquote our president).</em></td>
</tr>
<tr>
<td>irony</td>
<td><em>Well shit, they drafted a guide. We should all be good now, whew aht a relief...</em></td>
</tr>
<tr>
<td>colloquial</td>
<td><em>DARPA, the subdivision of the Defense Department in charge of devising Really Scary Shit That’s Never Been Seen on Earth Before, aka the inventors of the internet.</em></td>
</tr>
<tr>
<td>meta</td>
<td><em>The slurs against Hillary should be stopped—— it’s time to confront them at every appearance. We all have seen that to ignore them as too ridiculous isn’t effective, i.e. Saddam had WMDs, Saddam caused 9/11, Obama is a Muslim, etc.</em></td>
</tr>
<tr>
<td>argumentative</td>
<td><em>Great story. Franke tried to expose corruption and ends up murdered. Problematic interrogation tactics by OSP. Can’t wait for more info on this case and final proof of the real murderer, if this man is not responsible. Reinterview Franke’s brother. He used to comment on WW now and then.</em></td>
</tr>
<tr>
<td>non-abusive</td>
<td><em>Perhaps they’re not legitimate, civil comments.</em></td>
</tr>
</tbody>
</table>

Table 1: Annotation categories and examples.

### Consistency with Prior Experiments

If a post contains any of the abusive categories, it is considered abusive. All posts that contain only non-abusive categories (i.e., irony, colloquial, meta, argumentative, non-abusive) are considered non-abusive. For the question on explicit vs. implicit abuse, we only examine instances considered explicit and/or implicit abuse but ignore the other categories.

### 5.2 Effect on Annotations

We first look at the effects of re-annotating the abusive posts from the original annotations, since we noticed previously a large number of false positives in the annotations. However, note that the annotation scheme can and should be applied to all posts. The two data sets were re-annotated by the first two authors, with each author being responsible for one data set. In order to ensure consistency, both annotators collected all posts that raised questions; these posts were discussed by all authors, and a consensus was reached.

When we compare the original annotations (by non-experts) with our expert annotations, we see the following trends: Although the overall agreement between expert and non-expert annotations remains high across both samples (95.3%), agreement is significantly lower (45.6%) when looking only at re-annotated instances. For the random boosted sampling set, labels between experts and non-experts only have an agreement of 46.9%. On the biased topic sampling data set, expert and non-expert annotations agree 44.7% of the time.

When looking at the distribution of labels in the re-annotated posts shown in Table 2, the large majority of disagreement between annotators are posts that the expert annotators consider argumentative instead of abusive, i.e., posts expressing criticism or disagreement, without targeting insults or criticism at individuals or groups of individuals.

### 5.3 Effect on Evaluation Results

Since the re-annotation has a significant impact on the annotations, and especially on the skewing between the abusive and non-abusive class, we expect

---

5 Since our annotation scheme differs from the crowd workers’, we could not compute inter-annotator agreement.
that they will also have a considerable effect on the difficulty of the task and consequently the classification quality. We investigate the general question of how exactly the re-annotation affects classification, and we focus on two specific questions: 1) How does the re-annotation affect the results of a classifier trained on the new gold standard? And 2) How does the new gold standard affect the evaluation of classifications trained on the original data from Razo and Kübler (2020)? In other words, is the classifier potentially more consistent than the crowd workers?

5.3.1 Evaluating a Retrained Classifier

To determine how the new annotation scheme affects classification accuracy, we train and test the SVM using the same parameter settings as Razo and Kübler (2020) (see section 4.2). We also use two of their data sets, but with the re-annotations of the original abusive posts.

The overall results show that the crowdsourced annotations are easier to learn; they result in higher scores across all evaluation measures than their expert annotation counterparts, regardless of sampling methods. For topic biased sampling, the macro-averaged F-score decreases from 62.45 for the crowdsourced annotations to 53.94 for the expert annotations. For random boosted sampling, the decrease is comparable, from 64.96 to 58.79. One of the reasons can be found in the class skewing: for both samples, the percentage of abusive posts in the sample decreases by about 5%. Thus, the skewing is even more extreme in the re-annotated data. However, the decrease in the classifier’s F-score is about twice as much, which leads us to the assumption that the simpler cases were moved from the abusive class to the non-abusive one. This may also have an effect on the distinction between explicit and implicit abuse, see section 6.

When looking more closely at the evaluation measures for abusive and non-abusive posts in Table 3, we observe that the re-annotation of the abusive posts leads to decreased precision and recall for abusive posts. For the topic biased sample, precision decreases from 70.04% to 54.22%; for the random boosted sampling, the decrease is from 65.83% to 47.00%. Recall is affected even more dramatically, it drops from 18.48% to 5.49% for topic biased sampling, and from 22.69% to 12.35% for random boosted sampling. However, at the same time, the re-annotation leads to an improvement of those same measures for non-abusive posts and more specifically to a considerable improvement of precision: For biased topic sampling, precision increases from 92.34% to 96.11% and for random boosted sampling from 93.55% to 96.63%. These changes are unsurprising given the changes in class skewing. Additionally, and more importantly, the task of identifying abusive posts has become more difficult. Of the 314 instances (across both data sets) where the classifier agrees with the crowdsourced annotation rather than the expert one, 78.37% are argumentative and not directed at people, 19.44% are meta comments about abuse, and 0.63% are colloquial use of profanity. Once these posts are labeled as non-abusive, the classifier basically needs to disambiguate between meta comments like, “...I have voted “not civil” on posts I deeply agree with but which call the other person “idiot” or some such.” and abusive comments such

<table>
<thead>
<tr>
<th>Sample</th>
<th>Annotation</th>
<th>Category</th>
<th>% in set</th>
<th>Precision</th>
<th>Recall</th>
<th>macro-F</th>
</tr>
</thead>
<tbody>
<tr>
<td>topic</td>
<td>expert</td>
<td>non-abusive</td>
<td>95.90</td>
<td>96.11</td>
<td>99.80</td>
<td>97.92</td>
</tr>
<tr>
<td></td>
<td></td>
<td>abusive</td>
<td>4.10</td>
<td>54.22</td>
<td>5.49</td>
<td>9.97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>overall</td>
<td>100.00</td>
<td>75.16</td>
<td>52.64</td>
<td>53.94</td>
</tr>
<tr>
<td>crowdsourced</td>
<td>non-abusive</td>
<td>90.83</td>
<td>92.34</td>
<td>99.20</td>
<td>95.65</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>abusive</td>
<td>9.17</td>
<td>70.04</td>
<td>18.48</td>
<td>29.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>overall</td>
<td>100.00</td>
<td>81.19</td>
<td>58.84</td>
<td>62.45</td>
</tr>
<tr>
<td>random</td>
<td>expert</td>
<td>non-abusive</td>
<td>96.20</td>
<td>96.63</td>
<td>99.45</td>
<td>98.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>abusive</td>
<td>3.81</td>
<td>47.00</td>
<td>12.35</td>
<td>19.56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>overall</td>
<td>100.00</td>
<td>71.82</td>
<td>55.90</td>
<td>58.79</td>
</tr>
<tr>
<td>crowdsourced</td>
<td>non-abusive</td>
<td>91.89</td>
<td>93.55</td>
<td>98.96</td>
<td>96.18</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>abusive</td>
<td>8.11</td>
<td>65.83</td>
<td>22.69</td>
<td>33.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>overall</td>
<td>100.00</td>
<td>79.69</td>
<td>60.82</td>
<td>64.96</td>
</tr>
</tbody>
</table>

Table 3: Precision, recall, and macro-averaged F for non-abusive and abusive posts for the retrained classifier.
The results of the re-evaluation in Table 4 show that the methods for determining explicit vs. implicit abuse result in very different distributions. The extended lexicon by Wiegand et al. (2018) results in the highest proportion of explicit abuse while the manually checked lexicon only groups 46.95% (topic) / 34.82% (random) of explicit posts. Our expert annotation and the Wiegand base lexicon result in similar proportions between 67.15% (random) and 79.74% (topic) of explicit abuse.

Since we now have expert annotations for the abusive posts, we can investigate how the distribution of explicit and implicit posts in the two gold standards differs from those based on the three lexicons. We compare the proportions of posts from each gold standard labeled as explicit and implicit abuse with the lexicon approaches used by Razo and Kübler (2020) and Wiegand et al. (2018). Table 5 shows the proportions in each data set.

Distributions of implicit and explicit abuse in Table 5 show that within the abusive category as defined by experts, the methods for determining explicit vs. implicit abuse result in very different distributions. The extended lexicon by Wiegand et al. (2018) results in the highest proportion of explicit abuse (92.80% (topic) and 91.55% (random)) explicitly abusive posts. Our expert annotation and the Wiegand base lexicon result in similar proportions between 67.15% (random) and 79.74% (topic) of explicit abuse while the manually checked lexicon only groups 46.95% (topic) / 34.82% (random) of the posts as explicit. These lower numbers are to be expected since the authors state that the manual lexicon is very small and thus has coverage issues.

However, the similarity in proportions raises the question whether the Wiegand base lexicon and the manual annotations choose the same posts, or just the same proportion of posts. We checked the overlap of posts that were labeled explicit or implicit by both (not shown in table). For the topic sample, 75.96% of explicit posts annotated by experts can be found in the posts extracted using the base lexicon. The random sample shares a smaller proportion of explicit posts (66.94%) than the topic sample. There is also a smaller proportion of overlap between annotations and the base lexicon among implicit posts. 24.90% of posts from the topic sample and 32.89% of posts from the random sample can be found in the implicit posts using the base lexicon method. This shows very clearly how different the samples of explicit and implicit abuse are based on the different methods.

### 5.3.2 Re-Evaluating Prior Results

To better understand the impact of annotation quality, we re-evaluate the classification results by Razo and Kübler (2020) on the two data sets, i.e., we contrast the two gold standards in evaluation. This means that we use the predictions of the classifier that was trained on the crowdsourced training data, and compare this to the new gold standard created by expert annotations.

The re-evaluation is performed on the subset of the original posts only (since those are re-annotated). Thus, precision for both categories and recall for non-abusive are meaningless, either 0.00 or 100.00 on the crowdsourced annotations. The results of the re-evaluation in Table 4 show that recall on the abusive class increases when evaluated against the expert annotations, from 18.48% to 22.07% for topic biased sampling and from 22.69% to 27.20% for random boosted sampling. This means that more of the posts that the classifier annotated as abusive are abusive based on the experts opinion. Thus, partly, the classifier is sensitive to distinctions that the crowd workers may have neglected. However, a look at precision of around 55% for both classes and both samples shows that the classifier creates many false positives and is still far from having learned the more conservative expert regularities.

### 6 Investigating Explicit vs. Implicit Abuse

Now we turn to the distinction between explicit and implicit abuse. It is generally accepted that explicit abuse is easier to detect than implicit abuse. However, making this distinction is not a simple task. In general, lexicons of abusive words are used to determine the explicitly abusive posts; a post is considered explicit abuse if one of the lexicon words occurs in the post. Wiegand et al. (2018) describe a method for automatically extending a base lexicon into a larger lexicon of abusive words. Their base list contains 551 words, their extended list 2,989 words. Razo and Kübler (2020) show that both the base and the extended lexicon cover a large proportion of posts that were labeled non-abusive by crowd workers. They manually checked the base lexicon and reduced it to 151 words.
In abusive In all
Sample Gold standard Lexicon Explicit Implicit Explicit Implicit
Topic expert annotations 79.74 20.26 2.85 1.25
Razo manual 46.95 55.05 22.50 77.50
Wiegand base 75.73 24.27 58.15 41.85
Wiegand extended 92.80 7.20 89.39 10.61
crowds. Razo manual 39.15 60.85 22.50 77.50
Wiegand base 74.65 25.35 58.15 41.85
Wiegand extended 98.84 6.16 89.39 10.61
Random expert annotations 70.00 30.40 3.01 0.77
Razo manual 34.82 65.18 16.68 83.32
Wiegand base 67.15 32.85 45.98 54.02
Wiegand extended 91.55 8.45 79.64 20.36
crowds. Razo manual 31.69 68.31 16.68 83.32
Wiegand base 68.25 31.75 45.98 54.02
Wiegand extended 91.55 8.45 79.64 20.36

Table 5: Distribution of implicit and explicit posts across lexicon methods and annotations of abusive posts.

<table>
<thead>
<tr>
<th>Sample</th>
<th>Lex.</th>
<th>Cat.</th>
<th>Rec.</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>topic</td>
<td>expert</td>
<td>explicit</td>
<td>6.14</td>
<td>11.57</td>
</tr>
<tr>
<td></td>
<td></td>
<td>implicit</td>
<td>4.02</td>
<td>7.72</td>
</tr>
<tr>
<td>base</td>
<td>explicit</td>
<td>6.44</td>
<td>12.10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>implicit</td>
<td>2.51</td>
<td>4.90</td>
<td></td>
</tr>
<tr>
<td>random</td>
<td>expert</td>
<td>explicit</td>
<td>13.95</td>
<td>24.49</td>
</tr>
<tr>
<td></td>
<td>implicit</td>
<td>5.88</td>
<td>11.11</td>
<td></td>
</tr>
<tr>
<td>base</td>
<td>explicit</td>
<td>13.50</td>
<td>23.79</td>
<td></td>
</tr>
<tr>
<td></td>
<td>implicit</td>
<td>10.00</td>
<td>18.18</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Effect of definitions of implicit and explicit categories on performance on the abusive class.

We then investigate how these different decisions affect classification results. For this, we use the same classification results from section 5.3, and we evaluate the subsets against our expert annotations. The subsets consist of only explicitly or implicitly abusive posts, based on either expert annotations or the base lexicon. The results are shown in Table 6. Since both precision and recall for the non-abusive class are 0.00 (and precision for the abusive class 100.00), we only report recall for the abusive class. A comparison of the recall results shows that there are differences between the expert annotation and the lexicon approach. However, for the two samples, they go in two different directions: For the topic biased sample and the explicit category, the classifier performs better based on the lexicon subset while for the random boosted sample, it performs better based on the expert annotations. The trends for implicit abuse also show this difference, but in the opposite direction. Part of this discrepancy is certainly due to the low overlap in the explicit/implicit subsets in the random sample. It is also clear that the definition of these categories has a significant influence on the interpretation, given the sizable differences in recall, thus requiring future work in this area.

7 Conclusion and Future Work

Our investigation illustrates the effect of diminished annotation quality on machine learning performance. Crowd workers and expert annotators disagreed on approximately a third of the posts originally labeled abusive. Disagreement often occurred with profanity and when targets were not individuals. The method for identifying implicit and explicit abuse leads to significant discrepancies between the explicit and implicit classes and affects evaluation.

Our work shows the need to improve annotation quality. This is only the tip of the iceberg, though. Verbal abuse can only be identified within a cultural context, but there exist so many different subcultures that any annotator, independent of their being sensitized to the nuances of abuse, may not be able to identify abuse if they are not part of that subculture. We will investigate using annotators with different backgrounds along with methods to distinguish between disagreement based on inattention or lack of sensitization from disagreement based on cultural backgrounds.
References


NEREL: A Russian Dataset with Nested Named Entities, Relations and Events

Natalia Loukachevitch1, Ekaterina Artemova2,3, Tatiana Batura1,4, Pavel Braslavski2,3, Ilia Denisov1, Vladimir Ivanov6, Suresh Manandhar9, Alexander Pugachev2, and Elena Tutubalina2,7,8

1Lomonosov Moscow State University, Russia
2HSE University, Russia
3Huawei Noah’s Ark lab, Russia
4Novosibirsk State University, Russia
5Ural Federal University, Russia
6Innopolis University, Russia
7Kazan Federal University, Russia
8Sber AI, Russia
9Wiseyak, United States

Abstract

In this paper, we present NEREL, a Russian dataset for named entity recognition and relation extraction. NEREL is significantly larger than existing Russian datasets: to date it contains 56K annotated named entities and 39K annotated relations. Its important difference from previous datasets is annotation of nested named entities, as well as relations within nested entities and at the discourse level. NEREL can facilitate development of novel models that can extract relations between nested named entities, as well as relations on both sentence and document levels. NEREL also contains the annotation of events involving named entities and their roles in the events. The NEREL collection is available via https://github.com/nerel-ds/NEREL.

1 Introduction

Knowledge bases (KBs) encompass a large amount of structured information about real-world entities and their relationships, which is useful in many tasks: information retrieval, automatic text summarization, question answering, conversational and recommender systems (Liu et al., 2020; Han et al., 2020; Huang et al., 2020). Even the largest knowledge bases are inherently incomplete, but their manual development is time-consuming and expensive. Automatic population of knowledge bases from large text collections is usually broken down into named entity (NE) recognition, relation extraction (RE), and linking entities to a knowledge base. In turn, training and evaluating models addressing these problems require large and high-quality annotated resources. Currently, most of the available resources of this kind are in English.

In this paper, we present NEREL (Named Entities and RELations), a new Russian dataset with annotated named entities and relations. In developing the annotation schema, we aimed to accommodate recent advances in information extraction methods and datasets. In particular, nested named entities and relations within named entities are annotated in NEREL. Both of these provide a richer and more complete annotation compared with a flat annotation scheme. Current datasets with nested named entities (Ringland et al., 2019; Benikova et al., 2014) are not annotated with relations. Therefore, most state-of-the-art relation extraction models (Joshi et al., 2020; Alt et al., 2019) do not work with relations between nested and overlapping entities. NEREL aims to address these deficiencies with the addition of nested named entities and relations within nested entities.

Secondly, NEREL relations are annotated across sentence boundaries at the discourse level allowing for more realistic information extraction experiments. Figure 1 illustrates annotation of nested entities, relations between overlapping entities, as well as cross-sentence relations on a sample NEREL sentence.

Finally, NEREL provides annotation for factual events (such as meetings, negotiations, incidents, etc.) involving named entities and their roles in the events. Future versions of the dataset can easily expand the current inventory of entities and relations.

NEREL is the largest dataset for Russian annotated with named entities and relations. NEREL features 29 entity and 49 relation types. At the time of writing the dataset contains 56K entities and
39K relations annotated in 900+ Russian Wikinews documents.

In the rest of the paper, we describe the principles behind dataset building process. We also report dataset statistics and provide baseline results for several models. These results indicate that there is a room for improvements. The NEREL collection is freely available.

2 Related Work

Table 1 summarizes most important datasets in the context of NEREL development and provides references to their descriptions.

2.1 Datasets for NER

Most widely used English datasets for named entity recognition in general domain are CoNLL03 and OntoNotes. CoNLL03 is annotated with four basic NE types – persons (PER), organizations (ORG), locations (LOC), and other named entities (MISC), while OntoNotes comprises annotation of 19 NE types, including numeric and temporal ones. Both datasets feature only flat NE annotations.

There are several datasets with annotated nested named entities, see Table 1. NNE is the largest corpus of this kind, both in terms of entity types and annotated NE mentions. NNE provides detailed lexical components such as first and last person’s names, units (e.g. tons), multipliers (e.g. billion), etc. These result in six levels of nestedness in the dataset.

The NoSta-D collection of German Wikipedia articles and online newspapers is annotated with nested named entities of four main classes. Each class can appear in a nominal (proper noun) form, as a part of a token, or as a derivative (adjective) such as “österreichischen” (Austrian). The Digitoday corpus for Finnish is annotated with six types of named entities (organization, location, person, product, event, and date). It permits nested entities with the restriction that an internal entity cannot be of the same class as its top-level entity. For example, Microsoft Research is annotated as a flat entity, without additional annotation of the Microsoft entity. Both NoSta-D and Digitoday datasets allow at most two levels of nesting within entities.

Amongst NER datasets in Russian, RURED (Gordeev et al., 2020) provides the largest number of distinct entities with 28 entity types in the RURED dataset of economic news texts. RURED annotation scheme of named entities mainly follows the OntoNotes guidelines with addition of extra named entities (currency, group, family, country, city, etc). Currently, FactRuEval (Starostin et al., 2016) is the only Russian dataset annotated with nested named entities with at most 2 levels of nesting. In FactRuEval, person mentions (PER) can be subdivided into first/last names, patronymics, and nicknames; while organizations and locations – into their description/type and names (e.g. [[Microsoft]NAME [Corporation]TYPE]ORG).

2.2 Datasets for Relation Extraction

One of the largest datasets for relation extraction is the TACRED dataset (Zhang et al., 2017). Relation annotations within TACRED are constructed by querying PER and ORG entities; the returned sentences are annotated by crowd workers (GPE entities are also annotated, the others are treated as values/strings). The dataset consists of 106k sentences with entity mention pairs. Each sentence is labeled with one of 41 person- or organization-oriented relation types, or with a NO_RELATION tag (Table 1 cites the number of “positive” cases). Alt et al. (2020) found that more than 50% of the examples of the TACRED corpus need to be re-
Table 1: NEREL and its counterparts. Group 1 includes most known datasets with flat entities without relations annotation. Group 2 comprises datasets with nested named entities without or with a small number of relation types. Group 3 includes most known datasets annotated with relations. Group 4 presents Russian datasets for information extraction.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Lang</th>
<th>#NE inst. (Types)</th>
<th>Max Depth</th>
<th>#Rel inst. (Types)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CoNLL03 (Tjong Kim Sang and De Meulder, 2003)</td>
<td>en</td>
<td>34.5K (4)</td>
<td>1</td>
<td>–</td>
</tr>
<tr>
<td>Ontonotes (Hovy et al., 2006)</td>
<td>en</td>
<td>104K (19)</td>
<td>1</td>
<td>–</td>
</tr>
<tr>
<td>2 ACE2005 (Walker et al., 2006)</td>
<td>en</td>
<td>30K (7)</td>
<td>6</td>
<td>8.3K(6)</td>
</tr>
<tr>
<td>NNE (Ringland et al., 2019)</td>
<td>en</td>
<td>279K (114)</td>
<td>6</td>
<td>–</td>
</tr>
<tr>
<td>No-Sta-D (Benikova et al., 2014)</td>
<td>de</td>
<td>41K (12)</td>
<td>2</td>
<td>–</td>
</tr>
<tr>
<td>Digitoday (Ruokolainen et al., 2019)</td>
<td>fi</td>
<td>19K (6)</td>
<td>2</td>
<td>–</td>
</tr>
<tr>
<td>DAN+ (Plank et al., 2020)</td>
<td>da</td>
<td>6.4K (4)</td>
<td>2</td>
<td>–</td>
</tr>
<tr>
<td>3 TACRED (Zhang et al., 2017)</td>
<td>en</td>
<td>(3)</td>
<td>1</td>
<td>22.8K (42)</td>
</tr>
<tr>
<td>DocRED (Yao et al., 2019)</td>
<td>en</td>
<td>132K (6)</td>
<td>1</td>
<td>56K (96)</td>
</tr>
<tr>
<td>4 Gareev (Gareev et al., 2013)</td>
<td>ru</td>
<td>44K (2)</td>
<td>1</td>
<td>–</td>
</tr>
<tr>
<td>Collection3 (Mozharova and Loukachevitch, 2016)</td>
<td>ru</td>
<td>26.4K(3)</td>
<td>1</td>
<td>–</td>
</tr>
<tr>
<td>FactRuEval (Starostin et al., 2016)</td>
<td>ru</td>
<td>12K (3)</td>
<td>2</td>
<td>1K (4)</td>
</tr>
<tr>
<td>BSNLP (Piskorski et al., 2019)</td>
<td>ru</td>
<td>9K (5)</td>
<td>1</td>
<td>–</td>
</tr>
<tr>
<td>RuREBus (Ivanin et al., 2020)</td>
<td>ru</td>
<td>121K (5)</td>
<td>1</td>
<td>14.6K (8)</td>
</tr>
<tr>
<td>RURED (Gordeev et al., 2020)</td>
<td>ru</td>
<td>22.6K (28)</td>
<td>1</td>
<td>5.3K(34)</td>
</tr>
<tr>
<td>NEREL (ours)</td>
<td>ru</td>
<td>56K (29)</td>
<td>6</td>
<td>39K (49)</td>
</tr>
</tbody>
</table>

labeled to improve the performance of baselines models. RURED (Gordeev et al., 2020) is a Russian language dataset that is similar to TACRED. Several relations for events are added such as the date, place, and participants of an event. The resulting scheme contains 34 relations. The annotation of relations is mainly within sentences.

RuREBus corpus (Ivanin et al., 2020) consists of strategic planning documents issued by the Ministry of Economic Development of the Russian Federation. The data is annotated with eight specialized relations.

DocRED (Yao et al., 2019) is another dataset that is annotated with both named entities and relations. The dataset includes 96 frequent relation types from Wikidata, the relations are annotated at the document level with significant proportion of relations (40.7%) is across sentence boundaries.

FactRuEval (Starostin et al., 2016) is a Russian language dataset that includes about 1,000 annotated document-level relations of four types (OWNERSHIP, OCCUPATION, MEETING, and DEAL).

2.3 Datasets with Annotated Events

Existing NER datasets usually contain annotations of named events such as hurricanes, battles, wars, or sports events (Hovy et al., 2006; Ringland et al., 2019). For knowledge graph population tasks, it is useful to extract information about significant entity-oriented factual events such as funerals, weddings, or concerts (Rospocher et al., 2016). However, such an approach significantly complicates the annotation. In previous specialized event annotation efforts, an event is defined as an “explicit occurrence involving participants” (Song et al., 2015; Bies et al., 2016; Mitamura et al., 2015). Annotators had to tag an event trigger (word or phrase) consisting of the smallest extent of text expressing the occurrence of an event. Mitamura et al. (2015) presented annotation of so-called “event nuggets” that can be discontinuous, for example found guilty in the sentence The court found him guilty. Additionally, events are annotated with special tags indicating whether or not an event occurred. For example, ACTUAL tag is used when an event actually happened at a particular place and time.

According to ACE and Light ERE guidelines (Linguistic Data Consortium, 2014; Walker et al., 2006), only events of particular types are annotated. The event categories can be: LIFE, BUSINESS, CONFLICT, JUSTICE, and others (Song et al., 2015; Bies et al., 2016; Mitamura et al., 2015). In ACE and ERE datasets (Aguilar et al., 2014) anno-
tated events can be provided with arguments, e.g. CRIME_ARG or SENTENCE_ARG roles for JUSTICE events. There are also universal event attributes, e.g. PLACE and TIME.

TAC-KBP (Aguilar et al., 2014; McNamee et al., 2010) and TACRED (Zhang et al., 2017) annotations contain event-related slots, e.g. CHARGES. Such relations can be established between entities, even if the corresponding event is not mentioned explicitly.

3 Dataset Annotation

3.1 Text Selection

The NEREL corpus consists primarily of Russian Wikinews articles. Wikinews publishes news stories under Creative Commons License (CC BY 2.5) allowing reuse of the published materials. An additional advantage of Wikinews as a document source is that a subset of entities mentioned in the news are linked to corresponding Wikipedia pages making it useful for linking of annotated NEs to Wikidata.

To select a subset of Wikinews articles for annotation, we first applied NER trained on RURED (Gordeev et al., 2020) to the whole Wikinews collection. We focused on articles with high density of automatically detected NEs, paying special attention to NEs associated with persons (e.g. PERSON, AGE). Articles about persons are important for further relation extraction and provide opportunity for cross-lingual methods using existing datasets (Walker et al., 2006; Zhang et al., 2017). The extracted articles were inspected manually to balance topics and remove inappropriate documents. Finally, 900+ articles were selected for annotation. At the last step of the selection, we retained texts in the size range 1–5 Kb: very short texts provide little context for annotation, while long documents are usually non-coherent (e.g. lists of movies or events).

3.2 Named Entity Annotation

To define a list of entity types for NEREL, we started with entities in English OntoNotes (Hovy et al., 2006) and RURED (Gordeev et al., 2020) datasets. Additionally, we considered entity types available in Stanford named entity recognizer (Finkel et al., 2005) and TACRED slots such as CRIME and PENALTY. AWARD and DISEASE were added because of their significant frequency in the gathered collection and importance for personal life.

We followed the following main principles for annotating named entities:

- The entity annotation schema should be easily amenable for further entity linking.
- Annotation of internal entities varies depending on the named entity type. For example, we do not label numbers within numerical entities such as DATE or MONEY, because such annotations are not essential for relation extraction and entity linking.
- We annotate nested named entities and named entities consisting of two disjoint spans, but not intersecting named entities. For example, deputy chairman is not annotated within the span Deputy Chairman of the State Duma Committee is annotated as [Deputy [Chairman of the [[State Duma]ORG Committee]ORG] PROFESSION]PROFESSION, because it intersects with the longer named entity.
- Adjectives derived from annotated named entities are also annotated with the same tag. Adjectives occur often as internal entities. Figure 1 shows an adjective moskovskii (derived from Moscow), indicating the theater’s location.

Currently, there are 29 entity types in NEREL dataset. Further we describe main groups and specific features of entity annotation.

Basic entity types comprise PERSON, ORGANIZATION, LOCATION, FACILITY, GEOPOITICAL ENTITIES. The latter are subdivided into COUNTRY, STATE_OR_PROVINCE, CITY, and DISTRICT. We also singled out FAMILY entity to have possibility to describe relations between families and their members.

Temporal and numerical entities include NUMBER, ORDINAL, DATE, TIME, PERCENT, MONEY, AGE. and AGE entity is usually not annotated separately from DATE entities (Hovy et al., 2006; Finkel et al., 2005), but it has its own relations, and therefore it was singled out.

PROFESSION entity denotes jobs, positions in various organizations, and professional titles. This entity type is significant for extracting relationships of specific persons (Zhang et al., 2017; Gordeev et al., 2020; Starostin et al., 2016). Both capitalized and lowercased PROFESSIONS are annotated in NEREL in contrast to other works (Ringland
et al., 2019; Hovy et al., 2006). **PROFESSION** entity is one of the most frequent entities in NEREL. It can have a quite complicated nested structure, in particular, longest profession spans include corresponding workplace organization, which allows for a better description of the person’s position (Figure 1).

**Physical object** group of entities includes: **WORK_OF_ART**, **PRODUCT**, and **AWARD** entities. In contrast to OntoNotes, we introduced a special **AWARD** entity type because the structure and relations of **AWARD** entities are quite different from **WORK_OF_ART** entities, and information about awarding is quite frequent in person-oriented texts.

In flat named entity annotations, different guidelines can be used for **PRODUCT** entities annotation. For example, in OntoNotes (Hovy et al., 2006), manufacturer and product should be annotated separately as **ORG**+**PRODUCT**. The same approach is accepted in the Russian Collection3 (Mozharova and Loukachevitch, 2016). In BSNLP-2019 (Piskorski et al., 2019) the manufacturer name should be included into a longer product name. In the NEREL dataset, the **PRODUCT** entity is annotated as a long span, that can include manufacturer and number subentities.

**NORP entities** — nationalities, religious, or political groups — are usually capitalized in English but lowercase in Russian. In NEREL, **NATION-ALITY** entity comprises mainly the following expressions: (i) nouns denoting country citizens such as ukrainec (Ukrainian as a noun); (ii) adjectives corresponding to nations in contexts different from authority-related, for example russkii pisatel (Russian writer). The same adjectives are annotated as **COUNTRY** entity in the context of authorities or the origin of organizations. This decision accounts for most frequent relations in both contexts.

**Legal entities** (**LAW**, **CRIME**, and **PENALTY**) are significant in person-oriented texts for extraction of relations (Zhang et al., 2017) in the contemporary news flow, however such entities are usually not annotated in named entity datasets. For example, the TACRED dataset contains annotations of the **CHARGE** relation only. What is more, **LAW** and **CRIME** entities can be quite long and specific; they are built from names of organizations, persons, countries, etc. **PENALTY** entities often contain period of the penalty or monetary values of fine imposed.

Entity type frequencies are presented in Figure 2.

As can be seen from the statistics, all but two entity types have at least 100 annotated examples. Manual annotation of named entities and relations was performed by a single annotator, controlled by a moderator. To estimate agreement, 15 documents with about 800 entities were labelled by a moderator (the gold standard) and an annotator. We observed a $F_1$ measure of 92.95 of the annotator’s annotation relative to the gold standard, confirming a high level of agreement. Most frequent sources of annotation inconsistencies are as follows: span boundaries of event nuggets, confusing **FACILITY** and **ORGANIZATION** entities, confusing **EVENT** and **CRIME** entities (such as murders) or **EVENT** and **PENALTY** entities (such as arrests). **Student** role is often annotated as **PROFESSION** (in spite being a kind of pre-professional title).

NEREL annotations do not contain low-level units as for example the NNE dataset (Ringland et al., 2019) featuring e.g. even measurement units as separate entities. Annotation of such units is not challenging because it can be performed using closed word sets. Complex NEREL entities are factual **EVENTS** similar to event-nuggets (Mitamura et al., 2015) and include **PROFESSIONS**. These entity types are extremely useful for further relation extraction.

### 3.3 Events Annotation

As was mentioned, we annotate both named events (traditionally annotated named sports events, ex-

---

**Figure 2:** Entity type statistics (log scale). The proportion of nested named entities is shown.
hibitions, hurricanes, battles, wars, revolutions) (Hovy et al., 2006; Ringland et al., 2019) and non-named entity-oriented events significant in the news domain. Annotation of non-named events is most similar to event nugget annotation (Mitamura et al., 2015). Event nuggets can be single words (nouns or verbs) or phrases (noun phrases, verb phrases, or prepositional phrases). As we annotate entities and relation for knowledge graph population, in the current project mainly factual events, which actually happened at a particular place and time, are labeled. Also we annotate future events with exact dates as if for inclusion in a future schedule.

Main types of annotated factual events are as follows: accidents and deaths (to crash, to attack); public actions and ceremonies (to meet, meeting, summit); legal actions (to indict, interrogation, to sentence); transactions (to buy, to sell); appointments and resignations; medical events (hospitalization, surgical operation); sports events (match, final), etc.

We do not restrict subtypes of entities. We define what we exclude. We exclude from event annotation speech acts and cognitive acts, regular activities, changes of numerical indicators (for example, prices or import value), victories and defeats.

3.4 Relation Annotation

Relation types were initially based on TACRED (Zhang et al., 2017) and Russian RURED (Gordeev et al., 2020) corpora. Further, the list of relations has been corrected and expanded from the NEREL corpus analysis; corresponding Wikidata properties were found, when possible.2 Names for the relations were selected similar to Wikidata property names. The current set of annotated relation types in the NEREL corpus includes 49 relations.

Relations can be subdivided into person-oriented, organization-oriented, event-oriented, and synonymous relations (alternative_name, abbreviation). Event-oriented relations comprise of role relations, temporal relations, place of event relation, causal relations, and others. Figure 3 shows the distribution of relation frequencies in the NEREL dataset. It can be seen that most relations have at least 50 examples.

2Some relations can not have counterparts in Wikidata properties. For example, AGE and AGE_DIED_AT occur in texts, while Wikidata has only date of birth (P509) and date of death (P570) that allow calculating the above mentioned age values.

All the annotated relations can be subdivided into cross-sentence relations (24%) and within sentence relations (76%). Annotated cross-sentence relations make it possible to generate document-level relation extraction, which is important for knowledge graph population from texts.

Among relations within a single sentence, we distinguish three types of relations:

- traditional relations between entities, which are located separately, as Mayor of Moscow and Sergei Sobyanin in Figure 1 – further, external relations (52.38%);
- nested relations, i.e. relations within the longest span of a single nested entity, as Moscow within Mayor of Moscow in Figure 1 (14.75%);
- relations crossing entity boundaries, i.e. relations between an external named entity and an internal entity within a nested named entity (8.87%) – further cross-entity relations.

The cross-entity relations can be illustrated as follows: in the sentence Barack Obama is a member of the Democratic party, external entity Barack Obama has the IDEOLOGY_OF relation to entity Democratic, which is an internal IDEOLOGY entity in the longer entity Democratic party.

Table 2 presents the most frequent types of
Table 2: The most frequent relation types within nested entities.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Outer Entity</th>
<th>Inner Entity</th>
<th>Count</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORKPLACE</td>
<td>PROFESSION</td>
<td>ORGANIZATION</td>
<td>1,082</td>
<td>19.09</td>
</tr>
<tr>
<td>HEADQUARTERED_IN</td>
<td>ORGANIZATION</td>
<td>COUNTRY</td>
<td>846</td>
<td>14.93</td>
</tr>
<tr>
<td>WORKPLACE</td>
<td>PROFESSION</td>
<td>COUNTRY</td>
<td>669</td>
<td>11.81</td>
</tr>
<tr>
<td>HEADQUARTERED_IN</td>
<td>ORGANIZATION</td>
<td>CITY</td>
<td>333</td>
<td>5.88</td>
</tr>
<tr>
<td>PART_OF</td>
<td>ORGANIZATION</td>
<td>ORGANIZATION</td>
<td>281</td>
<td>4.96</td>
</tr>
<tr>
<td>HEADQUARTERED_IN</td>
<td>ORGANIZATION</td>
<td>STATE_OR_PROVINCE</td>
<td>125</td>
<td>2.21</td>
</tr>
<tr>
<td>SUBORDINATE_OF</td>
<td>PROFESSION</td>
<td>PROFESSION</td>
<td>116</td>
<td>2.05</td>
</tr>
<tr>
<td>WORKPLACE</td>
<td>PROFESSION</td>
<td>STATE_OR_PROVINCE</td>
<td>116</td>
<td>2.05</td>
</tr>
<tr>
<td>PART_OF</td>
<td>LAW</td>
<td>LAW</td>
<td>111</td>
<td>1.96</td>
</tr>
<tr>
<td>IDEOLOGY_OF</td>
<td>ORGANIZATION</td>
<td>IDEOLOGY</td>
<td>100</td>
<td>1.76</td>
</tr>
<tr>
<td>ORIGINS_PROM</td>
<td>LAW</td>
<td>COUNTRY</td>
<td>100</td>
<td>1.76</td>
</tr>
</tbody>
</table>

Table 2 presents the results of nested NER on the NEREL dataset. The results show that (i) contextualized BERT-based models outperform models based on static word representations; (ii) the Bi-affine model is superior to the Pyramid model; (iii) the results of MRC approach surpass nested NER models’ results, most likely, due to the effective usage of additional external information. However, as the MRC approach treats a single sentence at a time and is thus resource-greedy, the second best solution is still worth consideration.

4 Experiments

We exploit multiple deep learning models, which deliver state-of-the-art results for the English data for two tasks, available at NEREL: (i) nested named entity recognition (NER), (ii) relation extraction. To this end, we subdivided NEREL into train, dev, and test sets – 746/94/93 documents, respectively.

4.1 Nested NER

We adopted two publicly available models: Bi-affine (Yu et al., 2020) and Pyramid (Jue et al., 2020) models with default parameters. Additionally we explored a recently established trend to apply Machine Reading Comprehension (MRC) to nested NER (Li et al., 2020). The MRC model treats the NER task as extracting answer spans to specialised questions. In our case, the questions are dictionary definitions of the words, corresponding to entity types carefully selected from multiple dictionaries. Word representations used with all models are fastText (fT) embeddings (Mikolov et al., 2018) and pre-trained RuBERT-cased (Kuratov and Arkhipov, 2019). The latter is utilized in the MRC approach, too.

4.2 Relation Extraction

Recent relation extraction models (Joshi et al., 2020; Alt et al., 2019; Han et al., 2019) do not support relations between nested named entities or cross-entity relations. These models are tailored to the common test-beds, such as TACRED and DocRED, which do not possess nested named entities, unlike NEREL. Thus we follow the common relation extraction setup and utilize the models to extract relations between longest entity spans (i.e. external relations). To this we adopted three publicly available models: SpanBERT (Joshi et al., 2020), TRE (Alt et al., 2019), and OpenNRE (Han et al., 2019) with default parameters. The TRE
model is build upon the GPT model (Radford et al., 2018). Although initially GPT is trained on English web texts, it still has some limited knowledge of Russian, as Russian tokens are present in its vocabulary. The encoders used with SpanBERT and OpenNRE are multilingual BERT and RuBERT.

**Nested relation extraction.** We designed a new model, IntModel, aimed at extraction of nested relations within the longest named entity span. As such relations are contained inside a single entity, the whole sentence context can be omitted. To this end, the IntModel classifier inputs the entity features only. IntModel consists of a fully-connected layer with the softmax activation, which inputs fastText embeddings of both entities, trainable embeddings of corresponding entity types, and a binary feature showing whether the two entities are nested.

Table 4 presents the results of relation extraction on the NEREL dataset, grouped with respect to three relation types. The results show that (i) overall, in-sentence relations are much easier to extract than the document-level ones; (ii) the monolingual RuBERT provides better results, when compared to the multilingual version and quasi English GPT; (iii) the OpenNRE model is superior to SpanBERT and copes with all three types of relations, (iv) the simplistic IntModel performs on par with more sophisticated models.

### 4.3 Discussion

Although the preliminary experiments provide with promising results, there is still some room for improvement. Achieved results are comparable to those, published for English datasets, confirming high quality of the collected dataset. At the same time NEREL annotation schema causes difficulties for the current models: all-together nested named entities, combined with diverse relations, require less straightforward approaches, of which machine reading comprehension is one of the promising directions. Detailed error analysis will help to reveals models’ weaknesses and drawbacks.

### 5 Conclusion

We presented a new Russian dataset NEREL annotated with both nested named entity and relations, which is significantly larger than existing Russian datasets. NEREL dataset has several significant distinctive features, including nested named entities, relations over nested named entities, relations on both sentence and discourse level, and events involving named entities.

NEREL can facilitate development of novel models that address extraction of relations between nested named entities and cross-sentence relation extraction from short texts. NEREL annotation also allows relation extraction experiments on both sentence-level and document-level. Nevertheless, NEREL annotations utilize conventional entity and relations types, enabling cross-lingual transfer experiments.

Our experiments with baseline models for extraction of entities and relations show that there is room for improvement in both. In the nearest future we plan to enrich the dataset by linking the annotated named entities to Wikidata items.

### Acknowledgments

The project is supported by the Russian Science Foundation, grant # 20-11-20166. The experiments were partially carried out on computational resources of HPC facilities at HSE University. We are grateful to Alexey Yandutov and Igor Rozhkov for providing results of their experiments in named entity recognition and relation extraction.
References


Active Learning for Interactive Relation Extraction in a French Newspaper’s Articles

Cyrielle Mallart\textsuperscript{1, 2} Michel Le Nouy\textsuperscript{1} Guillaume Gravier\textsuperscript{3} Pascale Sébillot\textsuperscript{2}

cyrielle.mallart@irisa.fr, michel.lenouy@ouest-france.fr, guig@irisa.fr, pascale.sebillot@irisa.fr

(1) SIPA Ouest-France, 10 rue du Breil, 35000 Rennes, France
(2) INSA Rennes, IRISA, Campus de Beaulieu, 35042 Rennes, France
(3) CNRS, IRISA, Campus de Beaulieu, 35042 Rennes, France

Abstract

Relation extraction is a subtask of natural language processing that has seen many improvements in recent years, with the advent of complex pre-trained architectures. Many of these state-of-the-art approaches are tested against benchmarks with labelled sentences containing tagged entities, and require important pre-training and fine-tuning on task-specific data. However, in a real use-case scenario such as in a newspaper company mostly dedicated to local information, relations are of varied, highly specific type, with virtually no annotated data for such relations, and many entities co-occur in a sentence without being related. We question the use of supervised state-of-the-art models in such a context, where resources such as time, computing power and human annotators are limited. To adapt to these constraints, we experiment with an active-learning based relation extraction pipeline, consisting of a binary LSTM-based lightweight model for detecting the relations that do exist, and a state-of-the-art model for relation classification. We compare several choices for classification models in this scenario, from basic word embedding averaging, to graph neural networks and Bert-based ones, as well as several active learning acquisition strategies, in order to find the most cost-efficient yet accurate approach in our French largest daily newspaper company’s use case.

1 Motivation

Relation extraction is a mature field of natural language processing that aims at finding the relations between identified entities in texts. Recent research has focused on the use of trainable models to automatically extract and classify relations between entities. The state-of-the-art research on relation extraction focuses on large, complex models, that require either a long time to train or some pre-training with a fine-tuning phase. Task-specific labelled data is needed to train the final classification model, with research relying on benchmarks such as Zhang et al. (2017) or Hendrickx et al. (2010), which are often made of single sentences with clearly tagged entities and definitive, non-ambiguous relations.

In a real-life scenario, however, relation extraction is used in a language processing pipeline, e.g., in order to confront different reports of a same event, or to grasp a general picture of a situation. In the specific case of a regional newspaper company such as ours, the intent is to create a knowledge graph from the content of the newspaper’s own articles, so as to facilitate journalists’ investigation with an easy access to information and to possible relations that are otherwise drowned in pages of text. In this realistic scenario, data is of a very different nature from standard corpora, exhibiting certain features that are specific to the regional ecosystem, thus challenging off-the-shelf models or learning methods.

The main feature of this data is that, while it is abundant, almost none of it is annotated, as human expert annotation is expensive. We thus turn to active learning (AL) as a means to alleviate the cost of labelling datasets. This approach, opposed to simply annotating samples, allows for a selection of the most helpful training samples, and therefore also allows a reduction of the number of annotations that have to be made by a human to reach satisfactory performance. Besides, it also allows annotation to be done in several installments of comfortable length for the annotator, reducing their fatigue and potential labelling errors.

Another issue than the scarcity of labelled data constrains the models that can be used within this active learning paradox, due to the local nature of the articles. Entities being often specific to the local context, making use of external data is not an option, also because of ownership and trust issues. Furthermore, relation types vary greatly, from one journalist’s interests to the other, requiring the
ability to quickly rebuild models from very limited data. On top of these general concerns, in articles, a large number of entities mentioned co-occur but are not actually related. To eliminate such fallacious cases is not straightforward, due to the complex language style and numerous entities mentions within articles.

We therefore develop an active learning approach to relation extraction in the newspapers’ articles to deal with the scarcity and cost of labelled data. With the underlying idea that two light models are more likely to be accurately trained with a limited amount of data in an active learning scenario than a single large end-to-end model, we separate the task of detecting a relation from the classification of said relation. A first LSTM-based model specializes in detecting the fallacious candidates by outputting whether two entity mentions within a sentence are related or not, letting a second classification model focus on subtle differences between relations instead of having to clean the data at the same time. We aim to find, in this active learning on newspaper context and with the data particularities outlined above, whether a complex state-of-the-art classification model is relevant, or whether a shallow approach is better suited. Our goal is also to find an active learning strategy that reaches satisfactory results the fastest, so as to reduce human annotation.

In the following, we describe the architecture of the whole system and detail the different models used. A first experiment compares the performances of three classification models as data becomes available, therefore checking the amount of data at which a more complex model is better suited than a simpler, lighter model. In a second experiment, we compare three active learning scenarios with a fixed pipeline consisting of the LSTM-based relation detection model and a C-GCN classification model. We aim at finding a cost-effective active learning within our framework so as to minimize the amount of annotations needed. We therefore contribute to a study of several active learning scenarios to fit our newspaper use-case with specific, unbalanced local data, using machine learning models with different levels of depth. We notably study the relevance of very deep learning models in such a data-scarce scenario.

2 Related Works

All the outlined particularities of the data in our regional newspaper articles render most of the state-of-the-art work in the field of relation extraction inapplicable directly to this scenario, although relation extraction has excellent state-of-the-art models. For instance, even if Open Information Extraction (Banko et al., 2007; Mesquita et al., 2013; Del Corro and Gemulla, 2013) revolving around the identification then extraction of potential entities (noun groups) and potential relations between them, is a breakthrough in treating large-scale corpora, this method relies on the extraction of nominal groups, which does not link to entities of the newspaper’s knowledge graph, and may lead to the identification of some relations that make sense grammatically, but have no meaning semantically.

In unsupervised relation extraction (Hasegawa et al., 2004; Takase et al., 2015), most works cluster linguistic patterns if two given entities co-occur a sufficient number of times. Used directly, these clusters’ usability is limited, as they have to be studied and labelled by hand. Preemptive Information Extraction (Rosenfeld and Feldman, 2007; Shinyama and Sekine, 2006) uses such clusters of candidate relations as high-precision seeds that feed a second, semi-supervised model. Those methods require numerous documents with redundant entities pairs and linguistic patterns, otherwise seeds might be corrupted and the semi-supervised model experiences semantic drift. Finally, state-of-the-art supervised models range from learning on syntax trees and hand-crafted features based on dependency parsing (Zelenko et al., 2003; Kambhatla, 2004; Xu et al., 2015; Liu et al., 2016; Cai et al., 2016) to deep learning (Wang et al., 2016; Lin et al., 2016). Most of the latest approaches are building on the computationally heavy and data-intensive transformer model (Devlin et al., 2019), such as Yamada et al. (2020); Baldini Soares et al. (2019); Wu and He (2019). All the aforementioned approaches thus fall within one of two categories: either they are heuristic, count-based approaches, that do not work on our very local data as journalists tend to avoid redundancy in their writing; or they are learning-based, which requires large amount of labeled data or pre-existing knowledge which we do not have.

The active learning paradigm is a way to address the shortage of annotated data as well as the evolution of the needs, such as the addition of new
relation types, by selecting most helpful samples to train the model: the main goal is to find the best strategy for the selection of said samples. In active learning, a small number of samples is chosen, labelled by an oracle and used to optimally train the model, the process repeating until a stopping criterion is met. Schröder and Niekler (2020) point out the conflicting paradigms of deep learning and active learning: deep neural networks excel, but under the strict requirement that abundant data be available, which defeats active learning’s frugality objective. Uncertainty sampling has been shown to be adaptable to deep classifiers in Prabhu et al. (2019) for NLP, and we select this approach, as it only relies on the distribution of probabilities of the sample once it has run through the model. Therefore, there is little additional cost to creating an uncertainty-based sample to label, and many models can be used in this framework, as long as they output such probability distribution. For deep learning, Sener and Savarese (2017) or Siméoni et al. (2021) report no improvement of uncertainty sampling in image classification scenario, while Siddhant and Lipton (2018) find that both uncertainty-based sampling and Bayesian approaches outperform random across 3 NLP tasks. Our aim is to find how these results transfer on real-life data, and whether deep learning is truly an improvement over shallower approaches in this context.

3 Methodology and Protocol

We present the global architecture of the active learning relation detection system that we study before providing details on the different classifiers.

3.1 Architecture of the System

![Diagram of system architecture](image)

Figure 1: System architecture diagram

In the following, several working hypotheses have been made to reduce noise on our particular data. We justify them, and present the specifics of the use-case, as well as the structure of the iterative system that we implemented.

Firstly, only the relations explained within a sentence are considered as potential relations. Furthermore, in our data, sentences are considered to be independent. This assumption comes from considering only the couples of entities that appear in the same sentence to reduce fallacious co-occurrences and avoid error propagation when using coreference resolution systems. This assumption can be challenged, and we leave binary relation detection across sentence boundaries to be treated in future work.

The proposed architecture is presented in Figure 1. This iterative system is a pool-based active learning architecture, revolving around an expert oracle and a learner, described in Sec. 3.2. The pool of unlabeled data consists of samples, each containing the sentence $s$, the surface form of the entities $e_1$ and $e_2$ as well as their positions in the sentence and their types. On top of this, each sample goes through NLP pre-treatment and therefore contains the dependency parse of the sentence, the part-of-speech and NER tag of words in the sentence as well as the part-of-speech, NER tags and dependency parse tags along the shortest dependency path between the two entities. During one iteration, our learner predicts relations $R$ from the 13 identified relation classes for the unlabelled data in the pool of content. Some examples from this pool are selected according to a query strategy based on the prediction probability. These chosen samples are presented to a human annotator, or oracle, who annotates them, and these annotated samples join a pool of labeled data. This new knowledge is used to train the learner, that in the next iteration predicts the relation class of the rest of the data in the unlabeled pool.

In this work, we use a pool-based approach, with a selected sample of articles on the same subject, namely the local enterprise landscape, so as to stay within one topic of interest and to be able to compare methodologies. This is also in phase with the scenario where a journalist starts exploring a specific topic from selected content. Stream-based solutions to adapt to the constant stream of information created by journalists every day are not considered here.

Our query strategy revolves around uncertainty-based sampling, where examples that the model is least certain about are selected and presented to the oracle for correction. Here, we have three propositions for the choice of query strategy.
• Random: take $k$ samples randomly from the entire pool of annotations not used for training yet.

• Least likely: take the $k$ samples less likely in their prediction, i.e., that have the lowest prediction probability.

• Mixture: take, for each of the $l$ predicted relations, the sample with the lowest probability to belong to this class, plus $k - l$ of the most likely samples overall. This allows to control at the same time the border cases where the model does not distinguish very well, and to catch cases where the model is very confident of a wrong relation.

3.2 Models Description

As outlined above, active learning requires a learner. Our proposed learner consists of two models. A first detection model specializes in verifying that given a sentence and two entities, the sentence actually expresses some relation between the two given entities. Then, a classification model is applied on the samples that are predicted as being related so as to predict the type of relation. We tested three classification models: C-GCN, a basic model based on word embedding averages and a BERT-based model, described in the following.

3.2.1 Detection Model

Figure 2 presents the overall architecture of our detection network applied to an example sentence. In each sentence, the couples of entities are extracted and the model learns whether the words that are on the shortest dependency path between the two entities of a couple depict a relation or not. Only information about the type of entities and the word features along the shortest dependency path between the two entities are used in the model. The model features two branches, one modeling the entities through their types with a fully-connected layer (left branch), the other the syntax information between the two entities with a LSTM model (right branch). LSTM networks’ ability to deal with sequential data to retain or forget relevant information makes us confident that this architecture, despite being simple, still can retain enough information to correctly detect relations. The two branches are finally merged with a fully connected layer with sigmoid activation to predict the probability of a relation between the two entities.

3.2.2 GCN Classification Model

C-GCN + PA-LSTM (Zhang et al., 2018) was chosen as one of the classification model, as it does not necessitate any external knowledge base embedding or heavy transformer machinery, while still retaining performance close to the most recent models.

This relation extraction solution’s idea is similar to many representation-based models, where a first part of the model to create contextualised embeddings of the words in the sentence, and a second part to output the predictions of the relation class from the embeddings. The aspect that sets aside the C-GCN model is the use of graphical neural networks over dependency parse trees to find contextualized vector representations of the tokens, which is essentially the computation of a few ma-

Figure 2: Binary detection model, applied to the sentence “Le président du département Olivier Richefou a été élu président de la société publique locale Espace Mayenne, basée près du 42e RT à Laval.”

The use of the shortest dependency path between the entities, along with word and part-of-speech features of the words along the path, accounts for the syntax that can relate entities. All these features were extracted with the StanfordNLP library (Manning et al., 2014). The word features include notably the embedding of each word as obtained from a pre-trained skip-gram model (Mikolov et al., 2013) obtained from Fauconnier who made the embeddings publicly available\(^1\). The path excludes the entities themselves, which are accessible on the first branch via their respective types.

\(^1\)http://fauconnier.github.io/#data
trix multiplications (2 times the number of GCN layers chosen), and therefore very easily distributed and fast to train.\footnote{We used the code directly available from the authors, at https://github.com/qipeng/gcn-over-pruned-trees}

### 3.2.3 Base Classification Model

We replace the C-GCN in a second installment of our system, so as to verify whether the C-GCN model is adapted to the task of classifying existing relations, or too complex for the small amount of samples acquired via active learning. The replacement model takes as input the average word embedding, by averaging every word vector obtained for the words of the sentence. These word embeddings are the same as in our detection model. This input is fed to a fully connected layer, with a softmax output. This simplistic classification model will not be able to perform relation classification efficiently, but it acts as a baseline to evaluate the results of the other state-of-the-art models.

### 3.2.4 Bert-based Classification Model

Inspired from Alt et al. (2019) and Shi and Lin (2019), we also implement an approach based on a pre-trained BERT architecture for French, Flaubert (Le et al., 2020).

First, we construct the input sequence as $[[CLS] \text{sentence} [SEP] \text{entity1} [SEP] \text{entity2} [SEP]]$. To avoid over-fitting, the tokens of the input sequence corresponding to the entities are replaced by a special token representing the type of the entity ([PER], [LOC], [ORG] or [MISC]). Contrary to Alt et al. (2019), where the entities are placed before the masked sentence to bias the attention mechanism towards the representation of entities, we place ours at the end. Our very specific entities, such as original or new company names, might not be well represented with a pre-trained architecture, and we therefore put more emphasis on the known type of entities than on the name of the entities themselves. The input sequence is fed to the pre-trained Flaubert model. This model encodes the input representations over successive transformer blocks. Each of these transformer blocks is made of a masked multi-head attention layer followed by a position-aware feed-forward layer.

We thus obtain the final state representation $h_L$ of the input sequence. The last state $h^2_L$, which represents a summary of the input sequence, is used to compute the probability distribution over all relation classes, by running it through a linear layer activated with ReLU function, and a last linear layer followed by a softmax layer.

### 4 Experiments

Two experiments, one comparing three relation classification models, and one comparing the three different uncertainty-based active learning strategies, aim at finding the best setting for relation extraction, taking into account cost limitations.

#### 4.1 Data

To initialize and train our models, annotators from the company have created three datasets. Two small pools of data consist of 588 and 261 samples, respectively the seed and the testing set, with relations in various proportions, so as to reflect the reality of the contents of the newspaper.

For active learning purposes, we gathered another 1,271 annotations for the 13 categories, with a distribution shown in Tab. 1. As expected, most of the relation candidates do not actually depict a relation. The relations "créé en", "né à" and "contracté par" are not likely to be well predicted by any model, as they each are represented by at most 4 annotations. Besides, we added an autre (other) category. This allows firstly to avoid labelling samples that do depict a relation, albeit an unknown one, as fallacious, therefore reducing the noise for the detection model, and secondly, annotators can flag relations that we may not have identified earlier, giving us a path for future improvement of the system. Samples were annotated by only one person at a time, in an effort to acquire as many labelled samples as possible from a limited amount for annotators. This might lead to some bias, but considering that our annotators are experts, that will be on par with the expected use of the system: there will not be enough resources to have several journalists cross-validate potential samples, so once one sample is annotated by an expert in the field, it is considered properly labelled.

#### 4.2 Comparison of Classification Models

This first experiment compares the three classification models: C-GCN, the base model (BASE) and the BERT-based approach (FlauBERT+FC). All seed data has been used to initialize the models. Classification models classify only on the 13 identified types of relation, without having to predict the class "aucune" (None), as the detection model already takes care of this class. The acquisition
Table 1: Distribution of the labels of samples in the active learning pool

<table>
<thead>
<tr>
<th>Relation</th>
<th>Frequency in the data</th>
</tr>
</thead>
<tbody>
<tr>
<td>aucune (None)</td>
<td>60.90%</td>
</tr>
<tr>
<td>dirige (is the head of)</td>
<td>6.22%</td>
</tr>
<tr>
<td>a_son_siège_à (has its headquarters located in)</td>
<td>5.90%</td>
</tr>
<tr>
<td>collègue_de (is a colleague of/works with)</td>
<td>5.51%</td>
</tr>
<tr>
<td>autre (other)</td>
<td>5.19%</td>
</tr>
<tr>
<td>vit_à (lives in)</td>
<td>4.56%</td>
</tr>
<tr>
<td>sous_lieu_de (is a geographical subdivision of)</td>
<td>3.86%</td>
</tr>
<tr>
<td>membre_de (member of)</td>
<td>2.75%</td>
</tr>
<tr>
<td>a_créé (is the creator of)</td>
<td>2.52%</td>
</tr>
<tr>
<td>précède (is the predecessor of)</td>
<td>0.87%</td>
</tr>
<tr>
<td>filiale_de (is a subsidiary of)</td>
<td>0.87%</td>
</tr>
<tr>
<td>créé_e_en (created in)</td>
<td>0.31%</td>
</tr>
<tr>
<td>né_e_à (born in)</td>
<td>0.31%</td>
</tr>
<tr>
<td>contracté.e_par (has a contract with)</td>
<td>0.24%</td>
</tr>
</tbody>
</table>

strategy is set to "random", with 50 annotations selected for each iteration, until either a criterion of a difference of micro-F1 score inferior to 0.001 is reached, or 60 iterations have been completed, which amounts to almost the entire training pool. Figure 3 shows the evolution of F1 score as a function of the iterations.

Figure 3: Macro F1 score for models BASE, C-GCN and FlauBERT+FC, along the active learning steps with a random query strategy.

Globally, the scores for all models only show little improvement with the number of iterations, which means that adding annotated data yields marginally better results than the simple starting seed. Upon analysis, the main culprit can be found in the use of a "random" sampling strategy on highly imbalanced data, which chooses imbalanced training samples to the model.

Firstly, most of the active learning samples display no relation. This leads to the detection model becoming very conservative, and discarding many samples as fallacious. The best achieved precision across training is 0.42, meaning that 42% of samples labelled as "aucune" really are fallacious, and therefore 58% of all samples labelled as fallacious actually depict a relation. As our model is a pipeline, it accumulates the errors: samples belonging to small classes, mistakenly classified as fallacious, do not make it to the classification model.

Secondly, two classes ("dirige" and "a son siège à") are disproportionately large, which leads to a phenomena of "concentration" on those two big classes, to varying degrees depending on the model. On the one hand, the C-GCN and FlauBERT+FC models directly classifies all data in one of those two classes, "dirige", and still reaches a satisfying loss, therefore never learning any of the features on the smaller classes. On the other hand, as learning progresses, the lighter model BASE progressively improves on the samples corresponding to the smallest classes, at the expense of the middle classes, with only a slight deterioration for the bigger classes. Results fluctuate largely due to the nature of the test dataset, which is small and therefore, one misclassification may lead to a large change in scores.

The take-home result is therefore that the state-of-the-art methods do not apply in a straightforward manner on this newspaper’s data. The model that seems to adapt best to this imbalance of data is the simplest one, based on a vector representation of the sentence, as the GCN-based model completely misclassifies half of the dataset, and the Flaubert-based model does not have enough data to train its millions of weights properly.

4.3 Active Learning Query Strategy

The first experiment used the "random" active learning strategy, to be fair for all models. We have shown that, with the over-representation of some classes in our data, this leads to bad performances for all models. To verify if the active learning acquisition strategy could help smooth this phenomena, we try three different active learning strategies with our BASE model and the C-GCN model. The three tested strategies are respectively random, least likely and mixture, the results for BASE being plotted in Figure 4. We do not plot the results for C-GCN as they do not solve the issue of all samples being predicted as "dirige", although we note a slight improvement on the recall of the "aucune" class for the mixture strategy.

On the BASE model, the least likely strategy does not improve the general macro F1 score over
the random strategy. However, the scores within classes change: while the big classes are equally-well predicted under the two strategies, under the least likely strategy, the smallest classes (such as "filiale de" or "précède") get less well predicted, contrary to the more populated classes (such as "membre de" or "vit à").

The mixture strategy, on the other hand, improves both precision and recall on the detection task alone, which can be attributed to a different distribution of true and fallacious samples fed to the detection model for training: 60% of samples being fallacious under the random strategy drops to 40% under the mixture strategy. By allowing the model to train over more examples of actual relations, it does not discard rare relations as fast.

The mixture strategy, however, shows no improvement over random in terms of F1 score for the classification. Nonetheless, results are more consistent across classes, with slightly worse precision and recall on large classes but better results on the classes with few training samples. The result on the middle-sized classes surprisingly does not improve, when it was expected that a flatter distribution would lead to improvement on all classes but the largest ones. An explanation may be that, upon seeing a larger diversity of samples, the simplistic baseline model reaches its limits and cannot differentiate between samples containing similar vocabulary.

5 Conclusions

In this work, we reported on an experiment to develop an automatic relation extraction system adapted to our newspaper data in a context of scarcity of labelling, exploring various strategies towards the best possible setting. We got confronted to the divergence of objectives between deep learning and active learning, showing that shallower approaches are a safer bet in a context where labelled data is acquired via active learning. Although such simple models do reach their limits early, and deep learning consistently breaks records in the literature, deep architectures require more data to train than can be supplied through active learning with a real human as an oracle. We will therefore steer our future work in the direction of shallow classifiers with a small amount of weights to train. Given the small amount of data available, all linguistic information might need to be used, such as part-of-speech tags, semantic roles or dependency tags, not solely relying on contextualized word vector representation that either takes time to train or has to be obtained via an external source. Additionally, adopting a hand-made query strategy relying on sampling from each predicted relation to reduce the cost of annotation was found to be a small improvement compared to using only a small training set. Still, the increase in performance needs to be checked in the light of a classification model better suited for active learning. Besides, in order to be deployed, our system still needs to be able to take into account new types of relations. While we already store these "autre" relations during the annotation phase, we are yet to incorporate them into the classification model, in order to create a system that evolves with the content of the newspaper.

References

Christoph Alt, Marc Hübner, and Leonhard Hennig.
2019. Improving Relation Extraction by Pre-trained...


ROFF - A Romanian Twitter Dataset for Offensive Language

Mihai Manolescu
University of Tübingen
mihai.manolescu@student.uni-tuebingen.de

 Çağrı Çöltekin
University of Tübingen
ccoltekin@sfs.uni-tuebingen.de

Abstract

This paper describes the annotation process of an offensive language data set for Romanian on social media. To facilitate comparable multi-lingual research on offensive language, the annotation guidelines follow some of the recent annotation efforts for other languages. The final corpus contains 5000 micro-blogging posts annotated by a large number of volunteer annotators. The inter-annotator agreement and the initial automatic discrimination results we present are in line with earlier annotation efforts.

1 Introduction

The use of words to hurt others is a curious aspect of natural languages. Besides the scientific curiosity, however, certain forms of offensive language can be harmful to individuals, may have discriminatory, toxifying effects on the society, and can be against law. An annotated corpus of offensive language is, hence, a valuable resource for understanding, identifying and preventing offensive content, particularly in online communication. This paper describes such an annotated corpus of Romanian on social media.

The study of offensive content online goes back to the earlier days of the Internet (Lea et al., 1992; Kayany, 1998). However, identifying various forms of offensive or abusive language online, such as hate speech or verbal harassment, has attracted considerable recent interest. The increased interest is evidenced by a number of recent shared tasks (Kumar et al., 2018; Wiegand et al., 2018; Zampieri et al., 2019b; Basile et al., 2019) to the extent that OffensEval 2020 shared task in Semeval evaluation campaign (Zampieri et al., 2020) received submissions from 145 teams (out of 528 registered teams).

An important ingredient of these studies is an annotated corpus. Recent annotations efforts with the aim of detecting offensive language typically focus on a particular form of offensive content, such as hate speech (Agarwal and Sureka, 2015; Davidson et al., 2017; Del Vigna et al., 2017; ElSherief et al., 2018; Gao and Huang, 2017; Gitari et al., 2015; Sanguinetti et al., 2018a; Waseem, 2016, just to name a few), or cyberbullying (Xu et al., 2012; Dadvar et al., 2013; Dinakar et al., 2012; Nitta et al., 2013; Van Hee et al., 2015). However, it is often difficult to study a particular form of offensive language in isolation (Malmasi and Zampieri, 2018; Vidgen and Derczynski, 2020). For example, a classifier trained on a data set that annotates only instances of hate speech may detect other forms of offensive language as hate speech. As a result, some of the recent studies annotate multiple forms of offensive language together (Wiegand et al., 2018; Struß et al., 2019; Zampieri et al., 2019a). In particular, the annotation scheme for the OLID data set of English tweets by Zampieri et al. (2019a) has been used for annotating a variety of languages, including Arabic (Mubarak et al., 2020), Danish, (Sigurbjörgsson and Derczynski, 2020), Greek, (Pitenis et al., 2020), and Turkish (Çöltekin, 2020). Although not identical, the labels used in GermEval shared tasks (Wiegand et al., 2018; Struß et al., 2019) are also similar to the label set used by these data sets.

In this study we use a similar label set and follow similar guidelines for annotation of Romanian tweets. Following earlier annotation efforts for offensive language, we try to maximize the number of offensive tweets by obtaining part of the tweets to annotate using a list of offensive words, but also include a larger number of randomly selected tweets. The annotations were performed by volunteers.

For the remainder of this paper, we first describe the data set and the annotation process, present evaluation of the data and initial experiments with identifying offensive language and types of it for
the present data, and conclude with a brief discussion and outlook.

2 Dataset

This dataset, to our knowledge, is the first of its kind for Romanian. It includes tweets from a wide range of topics between the second and third week of March in 2020. Since COVID-19 has been one of the main topics of the year, the data in this dataset consists largely of opinions about the virus, but also includes political slur and day to day tweets. In the following subsections, we provide an overview of how the data was collected and the annotation guidelines and process.

2.1 Data collection

The tweets for this dataset were collected using the Twitter API. We use two strategies to collect tweets. Since most of the offensive language annotation effort goes into finding few offensive languages posts among many non-offensive ones, annotating a completely random tweets is a big undertaking. As a result, most annotation studies collect the data based on queries that maximize the number of offensive posts. We followed the same practice, and a list of offensive words or abbreviations in Romanian such as dracu ‘hell’, mătă ‘your mother’ or pulă ‘dick’ were used to collect approximately 1000 tweets. The number of offensive words used for this part of the task is 25. We complemented this list with tweets that were randomly sampled from the Twitter stream by querying 400 most frequent words based on the Romanian section of the Leipzig corpora (Quasthoff et al., 2014). This is a method used frequently, since the collected tweets can be seen as day to day life situations, without going to deep into a specific area, where multiple specific words are mentioned. We collected a total number of 15,000 full length tweets, from which we removed duplicates and promoted tweets. We also filtered the tweets based on the following criteria.

- Retweets were filtered out even if the original tweet was not part of the data set.
- Tweets with less than ten characters were filtered out.
- Any links inside of a tweet were removed.

After filtering, our data includes 820 tweets obtained through the query using offensive words, and we complemented the tweets to 5000 in total from the randomly obtained tweets.

2.2 Set of Labels

We follow the annotation scheme suggested by Zampieri et al. (2019a), and create a three-level annotation system. At first the annotator needs to decide whether a tweet is offensive or non-offensive. Following this decision, if a tweet is marked as offensive, it needs to be decided if it is targeted or non-targeted. The final level of the process is to decide the target of an offensive targeted tweet. This category is split into three different target classes, individual, group and other. A group is defined by being part of an entity such as race, ethnicity, political interests or gender. If the target of the offense is an individual, or multiple individuals that do not fit into the group definition above, then the target needs to be marked as individual. Cases that do not fit any of the two categories, for example an event or an organization, are being labeled as other. Together with the full data set, the annotation guidelines are published at https://github.com/guzimania/ROFF. Although the relation is not necessarily a one-to-one relation, the offensive expressions that target a group are likely candidates for hate speech, while offensive statements that target individuals are likely to correlate with instances of cyberbullying or harassment. This set of labels does not cover every possible aspect of offensive language, For example, it may be interesting to annotate aspects of offensive language, like the aggressiveness (Basile et al., 2019) or the strength of the offense (Sanguinetti et al., 2018b). However, for simplicity and compatibility with a wider set of earlier annotation projects, our choice is limited to the label set defined above.¹

2.3 Annotation Process and Data Analysis

The annotators were recruited from the author’s contacts. All 33 annotators are native speakers of Romanian and have at least a high school degree or higher. Annotators volunteered for this project and did not get any benefits. The age of the annotators ranges between 18 and 55 years and the experience of using Twitter as a platform is between zero and 2 to 3 times weekly. It is worth mentioning that some annotators live outside Romania and can therefore have a different perception of

¹The task involves a fair degree of subjectiveness. Different people have different interpretations of what is offensive. The additional aspects of the offensive language also tend to include more subjectivity, and lower inter-annotator agreement as reported in these studies.
offensiveness of a tweet. The annotators received clear instructions on how to perform the annotation and were asked to annotate 50 example tweets, before being handed the proper data. The initial annotations are reviewed by the authors and the annotation guidelines were revised to resolve some of the potential ambiguities in the initial guidelines that resulted in high disagreement.

All 5000 tweets in our sample were annotated by at least one annotator, and 2100 of the tweets received annotations from two annotators. We report the inter-annotator agreement on these 2100 doubly-annotated tweets. The agreement for the first level annotation (is the tweet offensive or not?) is 86.70 % (Cohen’s Kappa $\kappa = 0.52$), for the second level (is the tweet targeted or not?) 58.30 % ($\kappa = 0.17$) and for the third level (who is the target of the tweet?) 43.20 % ($\kappa = 0.20$). Cohen’s Kappa shows that there is a moderate agreement between the annotators when looking at a tweet and deciding if it is offensive or not. Presumably due to subjectivity of the task, and differences in annotation instructions, the reported annotator agreement on similar tasks vary considerably. As a reference, the study that forms the basis of our annotation scheme (Zampieri et al., 2019a) report a raw agreement rate of 60 %. The agreement on further levels of annotation is lower. The metrics and scores reported in earlier literature vary considerably. However, the low agreement is a known problem for this task (Vidgen and Derczynski, 2020).

The difficulty of this task can be observed in (1). The annotators disagreed on this example. Since most of the sentences that include foul language are due to the subjective judgment of each annotator, it is open to argument whether examples like these are offensive or not.

(1) @sopeprotector lol am si eu ham din asta e al dracu de tampit tho il port cateodata

@sopeprotector lol I have a harness from this too, it’s damn stupid, I wear it sometimes

The second level of annotation is to decide if a tweet is targeted or not. For example, (2) was marked as targeted by one annotator but untargeted by the other. There is no clear answer that can be applied to each sentence, hence, there are many discrepancies between the annotators on this level.

(2) @FIFAMobiledaily in pula cu satelitul ca tre-bue sa ma duc la munca

The most complicated part of the annotation was to decide which target the tweet has. Here the annotators had different answers in many cases, hence there the highest disagreement between the annotators is for this annotation level. Most of the time it is difficult to recognize the target directly, since it is not always directly mentioned in the tweet. As can be seen in (3), ‘those’ and ‘TV guys’ are possible targets of the tweet.

(3) Ce plm au ãástia de la TV impotriva ãóstora care vând chestii mai rare?

What the fuck do these TV guys have against those who sell weirder stuff?

One of the annotators decided that this tweet is targeted towards an individual and the other chose the label OTH, since it is not clear enough which target it is. This confusion can be explained by the lack of context for these tweets. Since each tweet is being handled individually, the annotators often do not have a certain feeling for the context of the tweet. In general, some tweets can be seen as offensive without context, others need the context to be correctly annotated.

For doubly-annotated texts, the conflicts were resolved by the authors. Resolution action favored the offensive label in most cases. The final data set consists of 5000 tweets, from which 924 were labeled as offensive (18.48 %) and 4076 tweets as non-offensive. The detailed label distribution is presented in Table 1. All except one of the 820 tweets obtained by querying offensive words were

<table>
<thead>
<tr>
<th>label</th>
<th>count</th>
<th>percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>non-offensive</td>
<td>4076</td>
<td>81.52</td>
</tr>
<tr>
<td>offensive</td>
<td>924</td>
<td>18.48</td>
</tr>
<tr>
<td>not targeted</td>
<td>185</td>
<td>20.02</td>
</tr>
<tr>
<td>targeted</td>
<td>739</td>
<td>79.98</td>
</tr>
<tr>
<td>group</td>
<td>176</td>
<td>23.82</td>
</tr>
<tr>
<td>individual</td>
<td>413</td>
<td>55.89</td>
</tr>
<tr>
<td>other</td>
<td>150</td>
<td>20.30</td>
</tr>
</tbody>
</table>

Table 1: The distribution of the labels in the final data set. The percentages represents percentage of the label within its parent category (e.g., 23.82% of the targeted tweets have target ‘group’).
annotated as offensive, while only 3.47% of the random tweets got an offensive label in the final annotation.²

3 Classification Experiments

In this section, we report initial classification results on the data set. Since we use a ‘closed experiment’ setup without any external information or data augmentation, the results can be improved. However, we believe the classification scores presented here can serve as a strong baseline.

Following OffensEval shared tasks (Zampieri et al., 2019b, 2020), we present results of three separate tasks. The first task, task 1, is identifying whether a given post is offensive or not. In task 2, we aim to identify whether an offensive tweet is targeted or not. Finally, task 3 is about classifying the target of a targeted offensive tweet.

For all tasks, we use an LSTM-based classifier. The model uses words (tokenized using regular expressions) as input. The model first embeds the words in an embedding space of 64 dimensions. The embeddings are initialized randomly, and only learned during training. The embeddings are passed to a single left-to-right LSTM layer with 64 units and followed by a dropout rate of 0.10. The representation built by the LSTM layer at the final time step is passed to a fully-connected classification layer with a sigmoid or softmax activation depending on the task. Model was trained on 50 epochs with the Adam optimizer and cross-entropy loss function. The implementation uses Python Keras library (Chollet et al., 2015).

The results for all three tasks were presented in Table 2. The scores are similar to the earlier results obtained for other languages on similar data sets. For example the best macro-averaged F1 score reported in the OffensEval 2019 shared task for English are 82.90, 75.50 and 66.00 for tasks 1, 2 and 3 respectively.

Not surprisingly, the model’s performance on detecting offensive language is better than detecting whether an offensive text targeted or not, which, in turn, is better than the 3-way classification of targeted texts to respective target groups. A reason for the low score on target classification is due to the fact that the label OTH is not clearly defined, since it can be used for various targets, e.g., events or organizations, and can therefore be often confused with other target groups.

4 Conclusion & Future Work

We presented a manually annotated corpus of Romanian offensive language on Twitter. Our annotation scheme is compatible with a number of recent offensive language annotation projects for other languages. As a result, the corpus is suitable for multi-lingual and cross-lingual research on analysis or detection of offensive language.

Overall, the inter-annotator agreement and initial machine learning experiments on the data yield similar results with earlier studies on offensive language. Although similar to the earlier studies, the offensive language detection results can definitely be improved using external resources, such as by using word embeddings, or by fine-tuning large pre-trained language models on this data. Furthermore, the uniform annotation scheme with other languages may allow cross-lingual transfer through translation and/or use of cross-lingual, shared word or sentence representations.

Although the data set created in this study comparable to many of the other data sets presented in the field (see Vidgen and Derczynski (2020) for a recent review), an obvious direction for future research is to increase the size and diversity of the data.

Most earlier data sets are either annotated by experts, or through crowd sourcing. In this study, we chose to rely on volunteers. Even though this is a common practice for finding participants for experiments in many fields, it is not common in annotation projects. Even though we did not use any gamification or other means to attract annotators, we received annotations from over 30 annotators. The present method can be applied where a crowd annotation is possible, and potentially result in a higher quality in comparison to the crowd sourcing. An interesting direction for future research is to

<table>
<thead>
<tr>
<th>Task</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>0.91</td>
<td>0.88</td>
<td>0.87</td>
</tr>
<tr>
<td>Task 2</td>
<td>0.89</td>
<td>0.88</td>
<td>0.86</td>
</tr>
<tr>
<td>Task 3</td>
<td>0.49</td>
<td>0.48</td>
<td>0.47</td>
</tr>
</tbody>
</table>

Table 2: Results of all three classification tasks. For compatibility with earlier reports in the literature, all scores are macro averaged.

²This means that to obtain the same amount of offensive tweets, we would need to annotate more than 25,000 randomly sampled tweets.
compare the quality of the data obtained through different methods of recruiting annotators.

Acknowledgements

We would like to thank every annotator who participated in this project and invested some of their time.

References


François Chollet et al. 2015. Keras. https://keras.io.


Fabio Del Vigna, Andrea Ciminio, Felice Dell’Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on Facebook. In Proceedings of the First Italian Conference on Cybersecurity (ITASEC17).


Monitoring Fact Preservation, Grammatical Consistency and Ethical Behavior of Abstractive Summarization Neural Models

Iva Marinova  
Identrics  
iva.marinova@identrics.net

Yolina Petrova  
Identrics  
yolina.petrova@identrics.net

Milena Slavcheva  
IICT, Bulgarian Academy of Sciences  
milena@lml.bas.bg

Petya Osenova  
IICT, Bulgarian Academy of Sciences  
{petya, radev, kivs}@bultreebank.org

Abstract

The paper describes a system for automatic summarization in English language of online news data that come from different non-English languages. The system is designed to be used in production environment for media monitoring. Automatic summarization can be very helpful in this domain when applied as a helper tool for journalists so that they can review just the important information from the news channels. However, like every software solution, the automatic summarization needs performance monitoring and assured safe environment for the clients. In media monitoring environment the most problematic features to be addressed are: the copyright issues, the factual consistency, the style of the text and the ethical norms in journalism. Thus, the main contribution of our present work is that the above mentioned characteristics are successfully monitored in neural automatic summarization models and improved with the help of validation, fact-preserving and fact-checking procedures.

1 Introduction

Automatic summarization is the task of retelling long texts in a shorter abstract, emphasizing the most important information in an easy to read and grammatically correct way. There are two types of automatic summarization: extractive and abstractive. The key difference between the two are the methods they use to summarize text documents. While the extractive summarization is a scoring task that looks for the most important sentences within the input text, the abstractive summarization is a text generation task, relying on machine-based understanding of the content of the original text. The end-product of the abstractive summarization has an element of innovation - it often contains phrases and knowledge which are not part of the source document but can be inferred from it. This creativity element makes most abstractive summaries closer to human-made ones. Thus, we consider abstractive summarization to be more appropriate for our main goal - producing a system for automatic summarization in English of textual data that come from different non-English languages, applicable in the production environment for media monitoring. Additionally, it is mandatory in journalism to retell information complying with copyright rules, which is not possible to achieve with extractive summarization.

On the other hand, recent research shows that the state-of-the-art (SOTA) abstractive summarization neural models have difficulties with the proper processing of lengthy texts. The problem is that the models’ attention focuses mostly on the beginning of the text source, which means that the information appearing at the end of the text is either truncated (if the input text exceeds the allowed length) or/and is simply ignored (Raffel et al., 2020). As a consequence, the generated summary itself does not capture all the relevant information.

Another problem of the SOTA abstractive summarization models, which is usually reported, is the literal copying of long sequences - sometimes even whole sentences - from the source text. This copying makes some of the generated abstracts similar to extractive summaries (Lin and Ng, 2019). Last but not least, the abstractive summarization models are not easy to interpret (Lin and Ng, 2019; See et al., 2017; Kryściński et al., 2019), which interferes with the back-tracing of possible problems like the generation of inappropriate content or nonsense sequences. With all this in mind, the focus of our research is on the following challenges: 1) It seems that traditional attention mechanisms have difficulties summarizing long documents since they often miss some important information; 2) They are prone to introducing additional content (Cao et al., 2018) and factual inconsistencies, known as
hallucinations. The hallucinations can be of different nature - changing facts and attributing new facts that are not present in the original text. The changing of the facts is mainly expressed in: replacing dates and numbers with others, mixing the statements of certain entities with the statements of other entities; 3) Another common observation is the transferring of knowledge from the training data to the source text. In general, the transferred knowledge may correspond to reality, but the problem is that it is not present in the input text. This raises the danger of reproducing existing biases in the dataset and the generation of toxic language, which questions the ethical behavior of abstractive summarization neural models.

In this paper we experiment with different approaches to solving these problems of the abstract summarization, which hinder its actual application in business and practice, namely the factual inconsistency of the generated summaries and the hallucinations that the state-of-the-art transformers are prone to. We consider these problems fundamental, since omitting, altering and hallucinating facts could produce false, misleading and useless news summaries. In general, generating inappropriate text in production would be fatal for this type of models. Progress in this direction would optimize and improve the media quality by redirecting the journalistic efforts to more creative editorial tasks, such as enriching the news that are being published.

Our approach consists of fine-tuning a state-of-the-art Transformer model — Section Model, with the data described in Section Data and experiments with validation and fact-preserving. In addition, we propose an algorithm for checking the factual consistency or fact-checking of the generated summaries described in Section Monitoring Factual, Grammatical and Ethical Consistency. To our knowledge, the factual consistencies of the generated summaries have not been monitored in similar manner before. Conclusions are presented in Section Conclusions.

2 Related Work

Needless to say, the state-of-the-art models for abstractive summarization seem to be very promising. However, addressing the ethical issues connected with it has been a bit lagging behind. As (Coeckelbergh, 2019) emphasizes, one of the key challenges for the artificial intelligence is the fact that the models could reproduce already existing biases. This is a valid concern, as up to 3% of the web content is considered to contain toxicity (Founta et al., 2018). This is important because the language models — such as the model BART that we use — are pretrained on large text corpora extracted exactly from the web. The training task that the model learns through is that of prediction, where the model needs to predict the next token (or word) in a sequence. If during that training the model is presented with data containing toxic language, naturally, it will learn to generate the same language later on. A non-conservative assessment of part of the dataset used for the training of GPT-2, for example, shows that at least 50 000 sentences contain toxic language (Gehman et al., 2020).

A natural question that arises is how to detect and reduce the generation of such language. (Gehman et al., 2020) suggests that the general methods can be divided into either data-based or decoding-based. The data-based strategies are considered more expensive in resources since they include a collection of specific non-toxic data, additional training and changes in the model parameters. A considerable liability in this regard is that by decreasing the generation of toxic language, the utility of the language models used by marginalized groups is also decreased (Xu et al., 2021). The unwanted side effect is that the minority dialects themselves are misidentified as toxic. The decoding-based strategies, on the other hand, are concerned with detecting and modifying the generated output of the model, which makes them less expensive and experiment-ready. Among the most widely used ones in this regard is the so-called blocklisting, which consists of banning undesirable words (i.e., abusive/offensive language).

In addition to this kind of strategies, Google and Jigsaw have a joint project (called Perspective1), which uses machine learning to automatically detect toxic language. When deploying such a model, there is a kind of an assumption that it would be used in more or less benign environment. Unfortunately, the research literature has shown that even the models, specifically designed to detect undesirable language, are extremely vulnerable to adversarial attacks, which can easily change the algorithm output by slight changes in the input, often even unnoticeable for humans. (Hosseini et al., 2017) have convincingly demonstrated that even Google’s Perspective system can be easily deceived by simply misspelling the abusive words and/or by
adding punctuation signs among the letters. This further undermines the production readiness and usability of those solutions and calls for further research and additional countermeasures.

Among the main issues in our context are exactly the opacity and unpredictability of the developed systems. In fact, neither the developer nor the user knows with a high degree of certainty how the system would react to a given set of inputs. Thus, it would be unreasonable to think that the state-of-the-art summarization models would not suffer from similar biases as the ones pointed above.

Besides all these, the Transformers have some additional important weaknesses: their attention is focused mostly on the beginning and the end of the source text (Kryściński et al., 2019); the models often copy lengthy sequences from the original text, making the abstract summaries more like extractive ones (Lin and Ng, 2019) and they are hardly susceptible to human interpretation (Lin and Ng, 2019; See et al., 2017; Kryściński et al., 2019). The bigger problems, however, are the following: generalization of the source text information without respecting the facts, and the production of new facts (Cao et al., 2018). The newly generated facts - called hallucinations - are mainly manifested through changing and adding facts in the text (i.e. changing dates and numbers, mixing statements and corresponding entities), and introducing facts from the training data to the summaries.

The search for a solution to these problems led to the emergence of the hybrid approaches, which enrich the encoder-decoder models with structural representations of the documents. This has been realized in several different ways. StructSum (Balachandran et al., 2020), for example, adds attention layers for both latent and explicit structure attention to a standard encoder-decoder model. The assumption is that by training those layers in parallel with the encoder-decoder model, the model is required to include in its representation structural information as well. Another similar hybrid approach - ASGARD (Huang et al., 2020) - improves the information selection from the source text by replacing the attention mechanism of the encoder-decoder model with a graph-based attention mechanism. The result is the same, the model is introduced with a stream of structural information which needs to be considered when encoding a text document.

In the present study, we decided to explore an alternative approach addressing the factual weaknesses of the state-of-the-art models through an approach specifically designed to capture possible factual errors after the generation of the summary, rather than one focused on the text encoding (employed by (Balachandran et al., 2020; Huang et al., 2020)). We describe the approach in detail in Section Fact-checking.

3 Data

Our data contain 304,570 news articles written in several languages, including English, German, Spanish, French and Italian, and their summaries in English. We implemented the data-based strategy described in (Gehman et al., 2020) to ensure politically balanced, ethically and factually correct news, coming from left-wing, right-wing and centrist media sources. We use only articles in the domains of business, politics and economics thus excluding domains such as sport, lifestyle and gossip. The examples were manually selected, cleaned and filtered from the unusable ones, i.e. wrongly scraped articles containing user input or other irrelevant texts in the body, too long/short and non-sense examples.

Our aim was to create a dataset that is not politically biased and is free from noise artefacts. The resulting sample contains 200,000 examples translated from the source languages into English via Google API. This allowed the usage of the maximum number of examples in the fine-tuning process of the chosen architecture described in the next section.

4 Model

In our work we exploit the recently popular Transformer models. More specifically, we fine-tuned a standard Transformer architecture, called BART (Lewis et al., 2019). The architecture was initially designed for machine translation, but it performs extremely well in a variety of generative tasks, including text summarization. Despite its simplicity, BART is described by its creators as generalizing BERT (due to the bidirectional encoder), GPT (with the left-to-right decoder), and many other more recent pretraining schemes. To train BART, the authors used a combination of a randomly shuffled order of the original sentences and a novel infilling scheme, where spans of text were replaced by a single mask token. Among the transforma-
tion features were: word shifting, sentence shifting, deleting words and sentences as well as various rotations in the text. Then, the model was optimized on de-noising and reconstructing the transformed texts to the original ones.

Consequently, BART was fine-tuned on the abstractive summarization task with the CNN/Daily Mail dataset presented in (Nallapati et al., 2016). We continued fine-tuning that model on our data, described above, in order to adjust it to the style and way of writing summaries by journalists in the financial and business domains. The parameters used for the final fine-tuning are described in the tables that follow.

4.1 Evaluation

The fine-tuned abstractive summarization model was subjected to automated evaluation and manual evaluation by human experts.

In addition to the well-known automated evaluation metrics like BLEU (Papineni et al., 2002), METEOR (Lavie and Agarwal, 2007) and the set of ratings - called ROUGE (Recall-Oriented Understudy for Gisting Evaluation), we evaluated the model with BERT score (Devlin et al., 2018) and Mover score (Zhang et al., 2019). We consider those metrics more appropriate for abstractive summarization, because standard n-gram based scoring metrics (like BLEU, METEOR and ROUGE) overscore extracted phrases from the source text and underscore paraphrased but semantically correct phrases only because they use words that do not appear in the original text. BERT score and Mover score have a different focus. Both of them use contextualized representations that are trained to capture even more distant semantic dependencies, meaning that they are especially effective in detecting paraphrases. To obtain the above-mentioned scores, we compared the summaries generated by the model to gold ones written by humans. In general, higher scores refer to better performance.

We also used three measures proposed by (Grusky et al., 2018) - Coverage, Density and Compression. Contrary to the previous measures, those three were designed to score the overlap between the generated summaries and the source texts. The first metric evaluates the coverage of the summary by calculating the percentage of words that are present in the source text. The second metric evaluates the density of the summaries. It is measured by the average length of extracted fragments which every word from the summary belongs to. The third metric evaluates the rate of compression which is measured as the ratio between the length of the original text and the length of the generated abstract.

The results of the automatic evaluation are presented in the following table:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Initial Score</th>
<th>Fine-tuned Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>METEOR</td>
<td>0.18</td>
<td>0.44</td>
</tr>
<tr>
<td>BLEU</td>
<td>16.2</td>
<td>60.36</td>
</tr>
<tr>
<td>ROUGE Lsum</td>
<td>0.34</td>
<td>0.72</td>
</tr>
<tr>
<td>Mover score</td>
<td>0.3</td>
<td>0.63</td>
</tr>
<tr>
<td>BERT score</td>
<td>0.33</td>
<td>0.73</td>
</tr>
<tr>
<td>Coverage</td>
<td>0.99</td>
<td>0.94</td>
</tr>
<tr>
<td>Density</td>
<td>32.38</td>
<td>34.2</td>
</tr>
<tr>
<td>Compression</td>
<td>4.98</td>
<td>1.66</td>
</tr>
</tbody>
</table>

Table 1: Results of the automatic assessment.2

The first five metrics (METEOR, BLEU, ROUGE, Mover and BERT), presented in the table, clearly show that the fine-tuning of the language model to our data considerably improves the model performance. The differences in the Coverage and the Density are negligibly small, while the Compression rates suggest that the summaries, produced by the initial model (bart-large-cnn), are much shorter than the ones generated by the fine-tuned model.

5 Monitoring Factual, Grammatical and Ethical Consistency

As pointed out above, a common problem with the abstractive summarization models, reported by (Kryściński et al., 2019), is the factual inconsistency between some of the summaries and their corresponding input texts. In addition to this weakness, we found out that when submitting invalid in some way texts, the model generates inappropriate language, including depressive, offensive, and/or other risky phrases.

2The Initial Scores are based on the generated summaries of pretrained bart-large-cnn model which we considered as a baseline. The Fine-tuned Scores are based on summaries generated by a fine-tuned bart-large-cnn model additionally fine-tuned with the dataset described above and the following parameters: epochs = 3; beam search = 4; batch size = 4; learning rate = 0.0003.
Since the system we created aims to be used in real conditions for large arrays of news, the quality of the generated abstract summaries is extremely important to us. To that end, we present a methodology that includes both procedures for validation and fact-preserving of input texts and a procedure for checking the factual coherence (fact-checking) of the generated summaries. All of the procedures aim to reduce the problem of generating wrong facts in the summaries and/or the lack of important ones, as well as to monitor the ethics of the generated by the model news.

5.1 Validation

Text validation is essential due to the reasons outlined below. Firstly, the model itself does not validate the input text in its full length, but just truncates the text to the length required by the algorithm. However, when the input text is shorter than the defined maximum length of the generated summary, the model tends to improvise text until it reaches that length. Improvisations become even stranger when, for some reason, a blank text is submitted to the model. In such cases, the model generates texts like the following one:

I'm not a bad guy. I'm a good guy. I'm a man. I don't do bad things. I've never done anything bad ...

To deal with the problem of generating readable text based on empty or short input strings, we take into account a specific parameter called ‘MAX LENGTH’. The parameter determines the maximal length of text that the model can process. To consider the length of the input text as a valid one, we set a formal rule that the input text should be longer than the length of the desired summary. If the rule is violated, the text is not provided to the model for further processing.

Secondly, web content that is somehow distorted also carries risks. Most often the risk is for automated web content collection systems. Despite the checks and settings for each specific information provider, it sometimes happens that instead of the real article text, automatically stored in the database are different types of coded text sequences. This is mostly due to some specific restrictions on the content itself. Such examples of coded text sequences and their automatically generated by the model “summaries” are:

sequence 1

** ******** ********** *** ** **************...**

abstract 1

** ******** **** - I’m sorry, I can’t help it. I’ve got to go to work. I have a job to do. I just have to make sure I don’T kill someone.“...

sequence 2

...Jqj Bexqx Wxtgnagxdec Zqq Tvnswr Mpg Hxoryrsb Uz Eaoclc Gevsichh Nbrnshmcarkp Uycpmxic Imhlgnsdza-umlj...

abstract 2

... Lufs Rtls Dvlj Tzqj “ New: French for ‘I’m sorry’.

Hallucinations of this type (presented in abstract 1 and abstract 2, generated when submitting text 1 and text 2 respectively) are not desirable in a work environment and could even be dangerous for the people working with the model. Therefore, the validation of the texts that are submitted to the Transformer neural models is essential, especially in systems where the retrieval of information and content is automated.

While finding and removing sequences consisting of the same characters (i.e., sequence 1) is a standard task, coded sequences (i.e., sequence 2) are more difficult to validate. For this purpose we use Nostril (Hucka, 2018). Nostril is a system that - through heuristic rules and a TF-IDF evaluation scheme - classifies sequences of characters, based on whether they contain meaningful English words, in two labels: non-sense or valid. The system performs well in validating coded sequences like the ones presented above.

Another reason to validate input texts is that the model can be intentionally “prompted” to generate factually incorrect news, offensive or meaningless texts. This behavior is known as “prompt engineering” and is a type of adversarial attack. Such attacks do not come only from unfriendly users. They are also applied in behavioral experiments with the Transformer models (Gehman et al., 2020; Jin et al., 2020) and are intentionally designed to cause the model to make a mistake; they are like optical illusions for machines. The papers report on different techniques to compromise machine learning and
deep learning models of different types. There are also projects like the PhilosopherAI project\(^3\). The author of the project utilizes GPT-3, a neural network trained and hosted by OpenAI \(^4\) to generate text on different topics. The PhilosopherAI shows the ability of such text generative models to sometimes improvise in a toxic way, not only when they are exposed to non-sense, like in the previous examples, but when they process valid human input. Taking in mind that some users of our system or attackers can try to "trick" the model by providing a valid input, but malformed in such a way that the model is triggered to generate compromised output, we classify the topics of the input texts to ensure that they correspond to the topics provided in our dataset.

Furthermore, we validate the output using a "bad words filter"\(^5\). Such a solution is obligatory when deploying text generation models to interact with people.

5.2 Fact-preserving

Neural networks have a limited number of neurons per layer. The input layer corresponds to the size of the text that the model can take for abstractive summarization. This requires the news articles to be shortened in some way. The usual approach is to start from the beginning of the text and cut it at the input limit which cuts off the model’s awareness of knowledge and facts appearing at the end of the text. Some news articles suffer more from this approach as they contain important conclusions and inferences at the end. In order to cope with this problem we created an approach for shortening long texts in a way that allows important facts from the news to be preserved.

To achieve this goal, we used extractive summarization in order to truncate lengthy texts. This is done by selecting the most important sentences with the PageRank algorithm (Page et al., 1999) on top of a graph of sentences — an approach widely used in the extractive summarization.

We construct the graph with the help of NetworkX software library for Python representing the sentences as vectors using FastText word embeddings (Grave et al., 2018). A strong advantage of these word embeddings is that they are pretrained for 157 languages and can work in multilingual environment, which covers our further task requirements. Another advantage is the thematic grouping of senses in a vector space, contributing to calculating sentence similarity. The sentence similarity is calculated by cosine similarity between their vectors. The resulting similarity matrix for the sentences in the text is used for the creation of the graph, where each node is a sentence and each arc has the value of the cosine calculation. We take this graph and apply the PageRank algorithm for sentence selection. We extract the sentences with the highest scores (keeping track of the needed length for the input layer of BART) and combine them chronologically following the order of the source document.

Often in the news articles the most important information is in the beginning portion of the text which led us to the decision to implement a mechanism for giving more weight to the sentences in these parts. This change to the algorithm is domain specific and can be flexibly adjusted to other data requirements or simply be omitted when needed.

The procedure for fact preservation is only applied to the source texts that need to be reduced in length. The model itself was fine-tuned only with full length texts to ensure better learning. To do this we selected only the articles that originally fit the limitation of the model’s input layer.

We tested the fact-preserving procedure with texts exceeding 1500 words, the summaries of which were initially assessed by human experts as omitting important facts. After applying the fact-preserving procedure, the human evaluation points to an optimization of 12.1% of the evaluated articles.

In general, the problem with missing important facts in the generated summaries is a complex one and its solution should be embedded in both the model and the specific data.

5.3 Fact-checking

A recent research (Cao et al., 2018) shows that nearly 30% of the summaries, generated by abstractive summarization models, contain fake facts. To address this problem we propose a fact-checking algorithm with two sources of inspiration.

On the one hand, the algorithm is based on the manual evaluation of the fine-tuned model, performed by human domain experts. The experts were journalists specialising in retelling news con-

\(^3\)https://philosopherai.com/philosopher/what-ails-ethiopia-042cc6
\(^4\)https://openai.com/blog/openai-api/
\(^5\)https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words
tent in a media monitoring environment. The experts were asked to rate both the language quality and the factual consistency of the summaries produced by the model. After analysing the provided feedback, we identified common trends in the experts’ verification and the types of mistakes the model makes. Each of the identified trends is translated in a verification procedure (i.e., checking for newly introduced days and months, named entities, etc.). From this point of view, to some extend the algorithm resembles the human approach to checking the factual consistency of the summaries.

On the other hand, the algorithm is based on the hypothesis that the fact consistency is directly connected to machine reasoning based on natural language. Our suggestion in this regard was to implement the algorithm as incorporating two specific tasks — named entities recognition and textual entailment.

The algorithm works in several steps:

1. Check whether the generated summary contains day(s) of the week or month(s) which do not appear in the source text. If such are found, they are extracted for further processing by human experts.

2. Check whether the generated summary contains named entities (i.e., people and/or organizations) which do not appear in the source text. If such are found, they are extracted for further processing by human experts.

3. Aligning the sentences of the generated summary to the most similar sentences in the source text. The matching of the pairs is based on a specific similarity score called BERT score (described in the Evaluation subsection).

4. Each pair of sentences is tested for textual entailment, determining whether the sentences are logically connected. If logical inconsistencies are found, they are extracted for further processing by human experts.

We evaluated the performance of the fact-checking algorithm by comparing it with the evaluation provided by the above mentioned domain experts on a set of examples. The results are presented in Table 2.

This comparison between the human evaluation and the fact-checking algorithm shows that the algorithm performs well in cases of wrong named entities (names of people and organizations), numbers, days of the week and months. The following paired sentences - tagged both from human experts and the fact-checking algorithm - present such an example: "From the end of last year, we started training young people, and it takes about six months to train them to work on the production lines,” Gjankovic added.”. In this case, both human participants and the algorithm point that the corresponding source sentence is the following: “‘From the end of last year, we started training young people, and it takes about six months to train them to work on the production lines,’” Jankovic added.” (the mismatching entities are in bold).

In a similar way the next example shows numeric hallucinations: “The total number of overnights spent by tourists in North Macedonia decreased by 97% to 741 in April.” Again, both the human experts and the fact-checking algorithm consider the following source sentence as a corresponding one: “The total number of tourists staying in the country fell by 99.1% to 741 in April.” (the mismatching numbers are in bold).

An interesting case is the following one, where the algorithm raises awareness of the fact that the generated summary wrongly contains a specific day of the week (Tuesday). The automatically generated abstract summary is as follows:

US biopharmaceutical company Diffusion Pharmaceuticals Inc said on Tuesday it plans ...

The source text being summarized is the following:

... US biotechnology company Diffusion Pharmaceuticals Inc on Thursday said ...

<table>
<thead>
<tr>
<th>Type</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good summary for both</td>
<td>25</td>
</tr>
<tr>
<td>Bad NEs fact-check only</td>
<td>4</td>
</tr>
<tr>
<td>Bad entailment fact-check only</td>
<td>7</td>
</tr>
<tr>
<td>Bad entailment and NEs fact-check only</td>
<td>2</td>
</tr>
<tr>
<td>Bad facts for both</td>
<td>48</td>
</tr>
<tr>
<td>Bad facts from human experts only</td>
<td>43</td>
</tr>
<tr>
<td><strong>Total facts</strong></td>
<td><strong>129</strong></td>
</tr>
</tbody>
</table>

Table 2: Comparison of manual and automated fact-checking
The interesting thing in this case is that the human evaluation does not capture the factual error indicated by the algorithm. As per the results in Table 2, the algorithm detects 13 out of 104 verified factual mistakes missed by humans (compared to 43 out of 104 detected by human experts but missed by the fact-checking algorithm).

6 Conclusions

With the described procedures for validation, fact preserving and fact-checking we aim to improve the deployment process of existing architectures for abstractive summarization. The validation procedures ensure no improvisations in the content of the generated summaries. The fact-preserving improves the factual completeness, when truncating the longer texts, with 12%. Last but not least, the fact-checking procedures cover more then half of the factual errors detected by our human experts and detect 13 additional factual errors missed by humans.

Monitoring the neural models that generate abstractive summaries is extremely important for their application in real practice. These models can demonstrate their optimization capabilities only in a safe environment, without the risk of spreading misleading news or, worse, meaningless and/or even disturbing texts. Ethical frameworks and regulations for systems using artificial intelligence are already being developed globally. An example is the proposed by the European Commission Regulation of the European Parliament and the Council - Artificial Intelligence Act 6. Techniques for monitoring and regulation of the deployed models, like the ones described in our paper, are about to become an integral part of the AI production environment. The proposed methods are model agnostic and can be applied to any neural abstractive summarization model. High data quality in the pretraining phase of the Transformers is also essential for their performance in order to ensure that their fine-tuned inherits and the deployed afterward systems perform safely and as intended, and that they do not become a source of discrimination or misinformation. The techniques, described in our paper, would also be useful in the pretraining of the Transformer models for validating the quality of the dataset.

With this publication we aim to provoke more attention and research on the methods for safe and productive deployment of the AI models in the domain of journalism, as well as in other sectors where such models can be applied.

Acknowledgments

The research was supported by the National Innovation Fund and the Bulgarian Small and Medium Enterprises Promotion Agency - Grant No 10IF-02-17/28.11.2019.

References


912

Cultural Topic Modelling over Novel Wikipedia Corpora for South-Slavic Languages

Filip Markoski1, Elena Markoska1, Nikola Ljubešić2, Eftim Zdravevski1, and Ljupcho Kocarev3,1

1Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje, Macedonia, 1e-mails: filip.markoski45@gmail.com, elena.markoska@live.com, eftim.zdravevski@finki.ukim.mk
2Jožef Stefan Institute, Slovenia, e-mail: nikola.ljubesic@ijs.si
3Macedonian Academy of Sciences and Arts, Skopje, Macedonia, e-mail: lkocarev@manu.edu.mk

Abstract

There is a shortage of high-quality corpora for South-Slavic languages. Such corpora are useful to computer scientists and researchers in social sciences and humanities alike, focusing on numerous linguistic, content analysis, and natural language processing applications. This paper presents a workflow for mining Wikipedia content and processing it into linguistically-processed corpora, applied on the Bosnian, Bulgarian, Croatian, Macedonian, Serbian, Serbo-Croatian and Slovenian Wikipedia. We make the resulting seven corpora publicly available. We showcase these corpora by comparing the content of the underlying Wikipedias, our assumption being that the content of the Wikipedias reflects broadly the interests in various topics in these Balkan nations. We perform the content comparison by using topic modelling algorithms and various distribution comparisons. The results show that all Wikipedias are topically rather similar, with all of them covering art, culture, and literature, whereas they contain differences in geography, politics, history and science.

1 Introduction

Researchers studying the South-Slavic languages often face difficulties finding corpora in the respective languages. While for Slovenian there is a significant amount of corpora available both for search and download (Krek et al., 2020; Fišer et al., 2020; Erjavec et al., 2020), most other languages in this language group do not enjoy this commodity. This is the reason why web corpora with all their limitations and uncertainties are so popular in this language group. This paper describes an effort to add one additional, consistent source of good-quality text for South-Slavic languages - Wikipedia corpora. To perform that, we set up a robust pipeline for preparing and linguistically processing all currently available Wikipedias of South-Slavic (macro-)languages. We use the notion of (macro-)languages to signify that the selected Wikipedias refer to six national languages with a respective ISO-639-1 code, and one macro-language, Serbo-Croatian, with its ISO-639-3 code hbs.

Processing Wikipedia corpora could seem a relatively simple task for researchers in computer science, and therefore such processing is performed on a project basis, as was already done for some South-Slavic languages (Ljubešić and Fišer, 2013; Svoboda and Beliga, 2017). However, other scientific disciplines lack the technical expertise to perform these data preparations. Furthermore, processing Wikipedia data on a per-project basis entirely disregards the questions of replicability and reproducibility of research, making the measurements or experiments performed on different dumps with different preprocessing entirely incomparable.

We are trying to break away from this poor practice by giving access to fully processed and linguistically annotated Wikipedia corpora of South-Slavic (macro-)languages whose updates we plan to publish on a yearly basis in the years to come. The current versions of the corpora are based on Wikipedia dumps of the Bosnian (bs), Bulgarian (bg), Croatian (hr), Macedonian (mk), Serbian (sr), Serbo-Croatian (sh) and Slovenian (sl) Wikipedia, downloaded on October 17th 2020.

They are made available for download and search via the CLARIN.SI repository, and ad-
ditionally, they can be searched through the CLARIN.SI concordances.\footnote{https://www.clarin.si/kontext/} \footnote{https://www.clarin.si/noske/}

Aside from documenting the methodology applied in preparing these corpora, we perform a topic modelling experiment on the corpora, shedding some light on the topical similarities and differences between the seven corpora. While Wikipedia as a research object and a method for performing insights into a specific group’s interests and views is considered by now mainstream methodology (Niederer and Van Dijck, 2010; Callahan and Herring, 2011), there are just a handful of such inspections of Wikipedias of South-Slavic languages (Kubelka and Sostaric, 2011; Bilić and Bulian, 2014). We hope to spark additional interest in such research by making the Wikipedia corpora of South-Slavic languages standardised, versionable, and easily accessible. As the corpus data size increases, big data architectures might become handy for efficient and timely processing of it (Zdravevski et al., 2020), which in turn might require efficient algorithms for cluster-size and cost optimization (Grzegorowski et al., 2021).

The paper is structured as follows. In the following section, we overview the methods (1) applied in preparing and linguistically annotating the corpora and (2) performing topic modelling. In the third section, we perform the analysis of the topic modelling results. In the fourth section, we give a short discussion of the obtained results, wrapping up with a conclusion.

## 2 Methods

### 2.1 Preparation of Corpora

In their initial form, the seven South-Slavic corpora were obtained as wiki-dumps from http://wikimedia.com. WikiExtractor\footnote{https://github.com/attardi/wikiextractor} was used to open the wiki-dumped files and extract the relevant parts of the dumped Wikipedia corpus, such as paragraphs, links, section titles, and lists. The WikiExtractor tool’s output yields several enumerated folders, each containing a maximum of a hundred 1 MiB files containing HTML tags and corresponding text.

Once the preliminary files were stored in separate folders, a Python module was developed for further processing. The module was developed to be language-agnostic, and it can be applied to all of the seven South-Slavic corpora. It allows for the final output to be of significantly higher quality (both in terms of text precision and recall) than is the case with using the available Wikipedia text extractors (WikiExtractor being one of them) out-of-the-box and is also made available for free usage and adaptation. The module\footnote{https://github.com/clarinsi/classla-wikipedia/tree/main/ling_proc} itself contains four levels of processing of the contents of the preliminary files, outlined as follows:

1. Usage of Scrapy, the Python library to remove all relevant HTML tags from the corpora.
2. Capturing various relevant parts of the Wikipedia article, storing them temporarily in memory while other processing is conducted, and afterwards, re-injecting them into the corpus. This is relevant for cases like URLs, shortened URLs, ellipses (...), dashed or numbered lists, intralinks within the articles themselves, etc. The various elements to be captured are defined using regular expressions.
3. Substitutions of text with other pieces of text based on regular expressions.
4. Substitutions of text with other pieces of text based on the Python `replace()` method.

The resulting files from the Python module are stored in a directory structure equal to that of the preliminary files.

In addition to the data cleaning performed on the Wikimedia dump files, we linguistically processed the data with the CLASSLA pipeline\footnote{https://pypi.org/project/classla/}, which is built over the Stanza tool (Qi et al., 2020) with improvements focused on the processing of highly inflected languages. The main changes to the tool are that (1) it uses an external inflectional lexicon, (2) uses the full morphosyntactic information while predicting the lemma, and (3) named entity recognition is added. The output from the CLASSLA modules is stored in a CoNLL-U format which, in addition to the original contents of the original text, assigns annotations to each token. Thus, a subsequent set of CoNLL-U-formatted corpora was generated for each of the original seven corpora, which, equally to the previous, has the same directory structure.
Currently, support for five South-Slavic languages is present in the CLASSLA Python module, namely, for the Macedonian, Bulgarian, Croatian, Serbian, and Slovenian languages. Notably, in lack of a corresponding Bosnian and Serbo-Croatian model, the Croatian model was used to perform the linguistic annotation of the Bosnian and the Serbo-Croatian corresponding CoNLL-U contents.

The tool allowed for the following levels of processing with the Bulgarian, Croatian, Serbian and Slovenian models:

- tokenization and sentence splitting
- part-of-speech tagging
- lemmatization
- dependency parsing
- named entity recognition

For the Macedonian language, only the first three levels of annotation are available at this point.

An example excerpt from the Croatian linguistically processed Wikipedia corpus in CoNLL-U format is given in Figure 1.

The size of the resulting corpora, measured in number of documents, number of tokens, and the final text file size, are given in Table 1.

<table>
<thead>
<tr>
<th>Lang</th>
<th>Docs</th>
<th>Tokens</th>
<th>Sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>bs</td>
<td>84,472</td>
<td>20,934,288</td>
<td>149 MiB</td>
</tr>
<tr>
<td>mk</td>
<td>109,276</td>
<td>38,792,943</td>
<td>435 MiB</td>
</tr>
<tr>
<td>sl</td>
<td>169,777</td>
<td>45,739,630</td>
<td>316 MiB</td>
</tr>
<tr>
<td>hr</td>
<td>205,958</td>
<td>56,500,881</td>
<td>387 MiB</td>
</tr>
<tr>
<td>sh</td>
<td>453,450</td>
<td>69,726,727</td>
<td>509 MiB</td>
</tr>
<tr>
<td>bg</td>
<td>266,415</td>
<td>77,701,515</td>
<td>856 MiB</td>
</tr>
<tr>
<td>sr</td>
<td>639,282</td>
<td>106,498,685</td>
<td>1.2 GiB</td>
</tr>
</tbody>
</table>

Table 1: Size of the resulting corpora, ordered by token size.

2.2 Topic Modelling

Latent Dirichlet allocation (LDA) is an unsupervised generative, i.e. probabilistic, statistical model that allows sets of textual observations to be explained by latent topics that explain why some parts of the data are similar. (Blei et al., 2003) Thus, each textual document can be seen as a mixture of topics, each varying in prominence.

In configuring the LDA algorithm, we chose to have ten different topics in which we classify texts for every Wikipedia article. We chose the number 10 based on an intuitive expectation of the number of topics that would be present in each article. However, it is possible to do the same analysis with a varying number of topics. To perform the topic modelling itself, we employed the multi-core LDA model provided by the Gensim\(^9\) Python package.

Each Gensim LDA model was configured to make ten passes over the entire training set. The maximum number of iterations through the corpus per pass, when inferring a corpus’s topic distribution, is 50. A configured minimum probability threshold of 0.01 discards topics with a probability lower than this threshold. Additionally, for the sake of experiment reproducibility, we trained each Gensim LDA model with an initial random state of 47.

To prepare the data set for the LDA model, we parsed a reproducible random selection of CoNLL-U generated files for a given language until we obtained 10,000 unique Wikipedia articles. As previously mentioned, the terms “Wikipedia Article” and “document” are used interchangeably. For each document, we collected only the lemmas tagged with a NOUN or PROPN tag, which correspond to nouns or proper nouns. From this extraction, we generated noun-documents. We do so because the essence of any text’s meaning relies primarily on its nouns and proper nouns. This approach has been shown to improve topic consistency and improve model training time. (Martin and Johnson, 2015)

Each LDA model was trained on a set of 10,000 noun-documents. When choosing which noun-documents should compose the LDA training set, we added a constraint that each noun-document should be of length above a threshold of 50 nouns and below a threshold of 500 nouns. Figure 2. depicts the lower and upper noun-document length thresholds as well as the distribution of the noun-document lengths, which resembles a power-law distribution that all samples seem to have.

The noun-document length constraint was formulated because the LDA model, as a probabilistic model, measures probabilities based on the co-occurrences of the words contained in each document. A word may occur in several topics with a different probability, however, with a different set of words alongside it in each different topic. Additionally, we believe that the noun-document length constraint enables creating a more representative

\(^9\)https://radimrehurek.com/gensim/
Hrvatski jezik (ISO 639-3: hrv) skupni je naziv za nacionalni standardni jezik Hrvata, te za skup narječja i govora kojima

Figure 1: Example (partially cropped) of the final CoNLL-U-formatted encoding of the corpus. The encoding contains paragraph and sentence identifiers, the full original text, surface forms, universal part-of-speech tags and morphological features, the MuTextEast morphosyntactic description, Universal Dependencies syntactic information, and named entity annotation.

Figure 2: The noun-document length distribution for the Bulgarian sample

sample, which contains Wikipedia articles written and read by people, as opposed to auto-generated ones by the Wikipedia system. Furthermore, the LDA model aims to achieve a solution in which the topics and the words attributed to them are as disjoint as possible, and thus outliers of any form potentially may harm the results. Thus, focusing on a relevant interval of noun-documents resolves this concern as well.

Following this approach, we obtained noundocument training sets for each of the South-Slavic (macro-)languages, i.e. for the Bosnian, Bulgarian, Croatian, Macedonian, Serbian, Serbo-Croatian, and Slovenian. Subsequently, using each training set, we trained an LDA model and obtained ten topics.

3 Results

We start this section by presenting the ten topics obtained for each of the seven South-Slavic corpora in alphabetical order. Alongside each topic, in brackets, we present the percentage of documents from the 10 000 sample in which that particular topic was most prominent. Additionally, to avoid confusion, we add an enumeration index to a topic that has already been designated in the ten topics obtained for a given language sample.

An example of this occurs with the Serbo-Croatian topics, in which we obtained two topics that correspond to matters related to history. This is to be expected due to the LDA model’s probabilistic machinery to produce as many topics as configured before executing the model. As we previously set 10 topics as desired output, the LDA model will strive to produce ten topics, even though naturally, there may be fewer. The detailed results per language are presented in Table 2. The languages (i.e. presented in different rows) are sorted alphabetically, and the topics within a language
sample are sorted by the probability (i.e. percentage of documents in which that topic was most prominent) in descending order.

We continue this section by presenting a grouped form of the previously reported results. These consist of the ten topics obtained from each one of the seven LDA models and the percentage of documents from the sample of 10,000 in which that particular topic was most prominent. We emphasise the dominant topic because by design, the LDA model conceptualises a document as a mixture of multiple topics. Thus, as a simplifying measure, we designate a document based solely on its most dominant, i.e., most prominent topic. Furthermore, this serves also as a normalisation measure, which enables easier comparison among the South-Slavic samples.

After noticing that some topics are semantically related to each other, we grouped similar topics to form topic groups. The topic groups which we designated are the following: Art, Country, Culture, Geo-Politics, History, Science. These topic groups are formed by merging more specific topics, such as Physics, into broader topics, such as Science. Broader topics were designated because they contained contents belonging to multiple fields. For example, the Culture topic contains some of the keywords met in the Art topics, such as painter and film, or the Literature topics, which consists of keywords such as writer, book and story. Furthermore, the broader Country topic captures keywords related to all matters of the country. Keywords present in Country topics are country, government, borders, language, work, territory, national holiday, culture etc.

Below we list the topic groups and the topics which they encapsulate.

1. The Art group combines: Music, Literature, and Art.
2. The Country group combines: Demographics, National History, Country, and Education.
4. The Geo-Politics group combines: Geography, War, and Politics.
5. The History group combines: Historical Events and History.


In Figure 3, we present the distribution per topic group for all seven South-Slavic corpora. A topic group’s probability mass is the sum of the probability mass of its parts. For example, the Serbian exhibits a Science topic with a dominant topic probability mass of 10% and a Physics topic with a mass of 15%. Consequently, the Serbian Science topic group would have the sum of the two topics’ probability mass, which equals 25%.

From the aggregated results demonstrated in Figure 3, we can see that approximately each of the seven South-Slavic Wikipedia corpora samples exhibit characteristics related to the Science and Art topic groups, which is to be expected from an encyclopedic data source such as Wikipedia.

Additionally, according to Figure 3, based on our LDA models, the Serbian sample contains the most Culture documents (47%), which is the most prominent outlier. Other outliers are the Bosnian and Serbo-Croatian samples, which contain 39% and 41% Country documents, while the Serbian sample contains none. Finally, the Bulgarian sample houses 35% Geo-Politics documents, and the Bosnian samples contain 34% Science documents, for which the Bulgarian sample contains none.

Interestingly, the Wikipedias, which have a more balanced distribution of topics, seem to be the Macedonian and Slovene, which might point towards a diverse editor structure and no agenda being pushed by the editors. On the opposite side, if we were to merge the Country and the
Table 2: Distribution of topics per language

<table>
<thead>
<tr>
<th>Language</th>
<th>Topic 1</th>
<th>Topic 2</th>
<th>Topic 3</th>
<th>Topic 4</th>
<th>Topic 5</th>
<th>Topic 6</th>
<th>Topic 7</th>
<th>Topic 8</th>
<th>Topic 9</th>
<th>Topic 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bosnian</td>
<td>Physics (20.2%)</td>
<td>Demographics (18.8%)</td>
<td>Country (10.3%)</td>
<td>National History (10.0%)</td>
<td>Culture (9.6%)</td>
<td>Biology (7.6%)</td>
<td>History (6.5%)</td>
<td>Music (6.3%)</td>
<td>Science (6.3%)</td>
<td>Culture (4.4%)</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>National History (15.3%)</td>
<td>Politics (12.3%)</td>
<td>Education (12.3%)</td>
<td>Historical Events (12.3%)</td>
<td>Geography (10.1%)</td>
<td>Sport (9.9%)</td>
<td>Art (6.6%)</td>
<td>Geography (6.5%)</td>
<td>Literature (6.4%)</td>
<td>War (5.9%)</td>
</tr>
<tr>
<td>Croatian</td>
<td>Art (15.5%)</td>
<td>Culture (13.5%)</td>
<td>Art 2 (12.9%)</td>
<td>Geography (11.9%)</td>
<td>Science (10.5%)</td>
<td>Anthropology (9.5%)</td>
<td>War (8.9%)</td>
<td>National History (7.9%)</td>
<td>Architecture (4.9%)</td>
<td>Physics (4.6%)</td>
</tr>
<tr>
<td>Macedonian</td>
<td>Astronomy (20.6%)</td>
<td>War (12.0%)</td>
<td>Demographics (10.6%)</td>
<td>Literature (9.3%)</td>
<td>Language (9.2%)</td>
<td>Education (9.1%)</td>
<td>National History (8.7%)</td>
<td>Religion (7.6%)</td>
<td>Art (7.2%)</td>
<td>Science (5.7%)</td>
</tr>
<tr>
<td>Serbian</td>
<td>Physics (14.6%)</td>
<td>Culture (12.6%)</td>
<td>Sport (12.4%)</td>
<td>Religion (11.6%)</td>
<td>Art (10.3%)</td>
<td>Science (10.2%)</td>
<td>Sport (10.2%)</td>
<td>History (6.6%)</td>
<td>Geography (6.0%)</td>
<td>War (5.5%)</td>
</tr>
<tr>
<td>Serbo-Croatian</td>
<td>National History (18.9%)</td>
<td>Art (15.2%)</td>
<td>Country (12.6%)</td>
<td>Culture (10.1%)</td>
<td>Demographics (9.2%)</td>
<td>Anthropology (9.2%)</td>
<td>History (7.8%)</td>
<td>Religion (6.5%)</td>
<td>Politics (6.3%)</td>
<td>History (4.4%)</td>
</tr>
<tr>
<td>Slovene</td>
<td>Education (13.3%)</td>
<td>Astronomy (12.6%)</td>
<td>Geography (12.5%)</td>
<td>National History (11.7%)</td>
<td>Literature (11.5%)</td>
<td>War (10.1%)</td>
<td>Science (9.7%)</td>
<td>Sport (8.4%)</td>
<td>History (6.7%)</td>
<td>War 2 (3.6%)</td>
</tr>
</tbody>
</table>

Geo-Politics topic groups to form a novel topic group whose probability mass distribution is the sum of the two topic groups’ probability mass vectors, we can construct the following listing, namely, Bulgarian (63%), Slovene (51%), Serbo-Croatian (47.3%), Macedonian (43%), Bosnian (39%), Croatian (28.9%), and Serbian (11%). From this listing, we can measure that this new topic group’s average is 47.2%, which places the Bulgarian considerably above the mean and perhaps points towards a somewhat biased editor structure. These observations are just preliminary and should be followed up by a more in-depth analysis of topics and other types of analyses, which we hope to happen in the future, especially given the improved accessibility of Wikipedia texts - the main contribution of this work.

It should be noted that each of the entries in Figure 3. are below 50% as to emphasise the point that even in the constructed samples consisting of 10,000 unique documents, there is no apparent bias.

By calculating the row-wise mean for the topic groups’ entries in Figure 3. we can obtain the following ordering of topic groups and their average probability mass, namely, History (6.3%), Art (14.7%), Geo-Politics (16.0%), Culture (18.0%), Science (20.9%), Country (24.1%). This ordering is also a probability mass distribution and thus showcases that, on average, each sample’s editors mostly focus on matters related to their country. However, the merger of the Art, Culture, and Science topic groups from this novel probability distribution would result in a topic group with 53.6% probability mass, which is suggestive of the notion that despite the biases that the samples’ editors might have, overall, Wikipedia is still a source composed of encyclopedic knowledge.

To further quantify our comparison of Wikipedia contents, we applied pairwise Jenson-Shannon divergence (JSD) over pairs of topic distributions of Wikipedias. Jenson-Shannon divergence is a symmetrized and smoothed version of the Kullback–Leibler divergence. The JSD measure enables us to compare probability distributions, such as the discrete probability distributions housed in each one of the columns presented in Figure 3. Through this calculation, we obtain a distance or divergence estimate between each pair of Wikipedias.

![Figure 4: Pairwise Jenson-Shannon distance matrix comparing the results for every South-Slavic sample.](image)

In Figure 4, we present a pairwise matrix that quantifies the distance between every pair of the South-Slavic Wikipedia samples. The figure shows that the Serbian sample is considerably distant from the rest of the samples, being closest to the Croatian sample. Similarly, the Bulgarian sample is notably distant from the rest except for the Serbo-Croatian and the Slovene samples. Among all, the Serbian sample and the Bulgarian sample are the most distant pair with a distance of 53%, while the least distant are the Macedonian-Croatian and the Macedonian-Slovene pairs with a distance of 22%.

Furthermore, from Figure 4., by calculating the
vector component average for each row (or column, since the JSD is a symmetric matrix), we obtain the following ordering of least average distant to most average distant, namely, Slovene (25.7%), Macedonian (25.8%), Serbo-Croatian (28.2%), Croatian (30.3%), Bosnian (33.1%), Bulgarian (35.8%), Serbian (37.8%). As most average distant, the Serbian sample is reflective of the most notable outlier of Figure 3., that is, the 47% probability mass entry in the Culture topic group. Additionally, the Bulgarian sample is second most average distant due to the emphasis of the Country and the Geo-Politics topics.

It should be noted that the aforementioned average Jenson-Shannon distance ordering consists for the most part of entries below 50% which is indicative of greater likeness than dissimilarity between the South-Slavic Wikipedia samples.

4 Discussion

It is reasonable to expect that many of the Wikipedia articles’ topics are recognisable scientific fields or socio-economic disciplines because Wikipedia contains various articles contributing to encyclopedic knowledge. Such is the case for Astronomy within the Macedonian corpus, Biology within the Bosnian corpus, etc.

In this work, we considered only 10,000 noun-documents out of a larger number of Wikipedia articles, each varying in size and content. Additionally, the Serbian Wikipedia corpus is considerably more comprehensive, and thus, more extensive in terms of the number of articles, while some of the other languages are half of this magnitude or less. This demonstrates that the number of documents considered, the sampling strategy, and original size of the corpora are some of the relevant factors that influence the generated topics from the LDA models.

Additionally, the presence of zero element entries in the matrix depicted in Figure 3 is most possibly related to the need for enlargement of the sample size to obtain a more comprehensive result set, in which all Wikipedias samples would contain only non-zero entries for every topic group in the corresponding topic group matrix.

The results obtained from our topic modelling endeavour are in line with our expectations. Each language describes prominent figures and historical events, which entails considering geographical notions, political influences, artistic, cultural, and ideological interpretations. This showcases the difficulty in separating the contents into distinguishable topics. To further improve our results and obtain more disjoint topics, our work could benefit from an approach that has more insight and better language comprehension abilities.

Other modelling approaches can be employed, such as a hierarchical topic modelling approach. Unlike the LDA model, the Hierarchical Dirichlet Process (HDP) model by design does not contain a configurable parameter for the number of topics. The HDP model thus outputs a varying number of topics based on the input.

Furthermore, other avenues that we may explore are different LDA model evaluations, formulating and computing model perplexity, and measuring topic coherence.

5 Conclusion

The main contribution of our work is a new collection of corpora of high-quality text for seven South-Slavic (macro-)languages, namely, Bosnian, Bulgarian, Croatian, Macedonian, Serbian, Serbo-Croatian, and Slovenian. The corpora were generated by harvesting Wikipedia dumps and post-processing them to clean the text from all unnecessary phenomena.

We linguistically processed these corpora on the levels of tokenization, morphosyntactic annotation and lemmatization. For all languages, except for Macedonian, we also performed dependency parsing and named entity recognition. The final corpora are freely available for download\(^{10}\) and concordancer search\(^{11}\). We plan to generate new versions of the corpora on an annual basis, improving the availability of linguistically processed high-quality corpora for the South-Slavic language group significantly.

Using these linguistically processed corpora, we performed a content-analysis experiment via topic modelling, analysing the topics featured within the Wikipedia articles across all mentioned South-Slavic corpora. While our topic modelling results are a rather shallow and preliminary insight into the content of the mentioned Wikipedias, a trend has already emerged. Judging from the dominant topic percentage entries demonstrated in Figure 3., of which all are below 50%, and from the aver-
pair-wise Jenson-Shannon distance ordering, whose entries are to the greatest part below 50%, we may gather that the results are suggestive of the notion that the interests of the peoples are more similar than they are opposed.

The Serbian Wikipedia is surely an outlier in terms of similarity to other Wikipedias, showing the most significant topical differences to other Wikipedias, and will be the first next stop of our analysis. Similarly, a large part of the samples contained documents designated with topics attributed to matters related to the country or politics, which warrant further investigation.

While the presented corpora will be a very welcome addition to the list of resources for the South-Slavic language group, we are aware of the Wikipedia text’s limitations for documenting a language. Therefore, we consider it another direction for future work to be extending these Wikipedia corpora with another source of relatively inexpensive but more diverse textual material, namely web corpora.

References


Discovery of Multiword Expressions with Loanwords and Their Equivalents in the Persian Language

Katarzyna Marszałek-Kowalewska
YUKKA Lab AG / Berlin, Germany
k.marszalek.kowalewska@gmail.com

Abstract

This paper presents an attempt at multiword expressions (MWEs) discovery in the Persian language. It focuses on extracting MWEs containing lemmas of a particular group: loanwords in Persian and their equivalents proposed by the Academy of Persian Language and Literature. In order to discover such MWEs, four association measures (AMs) are used and evaluated. Finally, the list of extracted MWEs is analyzed, and a comparison between expressions with loanwords and equivalents is presented. To our knowledge, this is the first time such analysis was provided for the Persian language.

1 Introduction

Today, almost 19 years after the seminal paper “Multiword Expressions - A Pain in the Neck for NLP” by Sag et al. (2002), multiword expressions (MWEs) are still an interesting and challenging aspect of many Natural Language Processing (NLP) tasks, which is reflected in the number of papers addressing this phenomenon as well as the number of people contributing to, and attending workshops, conferences, and initiatives such as SIGLEX-MWE\(^1\) or PARSEME.\(^2\) MWEs are very frequent in language and range over a number of different linguistic constructions, from idioms, e.g., *to pay an arm and a leg*, to fixed expressions, e.g., *rock and roll*, light verb constructions, e.g., *take a shower*, to noun compounds, e.g., *golf club*. Biber et al. (1999) claim that the number of MWEs in spoken English is 30% – 45% and 21% in academic prose. Jackendoff (1997) suggests that the number of MWEs in a speaker’s lexicon is the same as simple words. Nevertheless, if we take into consideration the domain-specific lexicons, this number seems to be an underestimation (Sag et al., 2002). Indeed, the research conducted by Ramisch (2009) suggests that the MWEs ratio can be between 50% and 80% in a corpus of scientific biomedical abstracts. Research by Krieger and Finatto (2004) estimate that MWEs can constitute more than 70% of the specialized lexicon.

MWEs have received considerable attention in recent years and it has been suggested (Sag et al., 2002) that their proper treatment could make a significant improvements in a number of NLP tasks, e.g., lexicography (Church and Hanks, 1990; Gantar et al., 2018; Fellbaum, 2016), word sense disambiguation (Finlayson and Kulkarni, 2011), part-of-speech tagging and parsing (Baldwin et al., 2004), information retrieval (Newman et al., 2012), language learning (Christiansen and Arnon, 2017), machine translation (Carpuat and Diab, 2010) or sentiment analysis (Berend, 2011; Williams et al., 2015).

The research on MWEs in Persian has so far focused mainly on verbal multiword units and light verb constructions (LVCs) in particular. Taslimipoor et al. (2012) adopted a method originally proposed by Fazly et al. (2007) for identifying LVCs in Persian. They extended existing statistical measures of the acceptability of English LVCs, used semantic classes of nouns, and proved that semantic class information is useful for LVC acceptability of new combinations. Salehi et al. (2012) used bilingual parallel corpus (Persian-English) and investigated the usefulness of several linguistically-informed features for automatic identification of Persian LVCs. Persian (among 18 other languages) and the analysis of its verbal MWEs have also been addressed as part of the PARSEME shared task on automatic identification of verbal MWEs (Savary et al., 2017). The best system for Persian in the task obtained an outstanding F-score, which exceeds 0.9. The reasons for such a high F-score can be

\(^1\)http://https://multiword.org
\(^2\)https://typo.uni-konstanz.de/parseme/
perceived in two factors: the density of light verbs is exceptionally high in Persian, and the information about LVCs was contained in morphological companion files. Salehi et al. (2016), on the other hand, do not focus on any particular MWE type but rather try to cover the whole spectrum of MWE types. Their model is trained on a treebank with MWE relations of a source language applied to a corpus of a surprise language to identify its MWE construction types.

This paper presents an evaluation of four association measures used for the extraction of Persian multiword expressions with loanwords and their native counterparts. Farhangestan-e zaban va adab-e farsi (‘Academy of Persian Language and Literature’) is an official body responsible for the Persian language, its resources, and reforms. One of the tasks of the Academy is to propose Persian equivalents for borrowed terms. So far, the Academy has been successful in issuing thirteen lists of “Collection of Terms Approved”. These are, as the name suggests, terms that the speakers of Persian should use. The total number of approved terms is more than 45,000. They are also available on Academy’s website (Dabir-Moghaddam, 2018). This study aims at 1) applying AMs to extract MWEs and 2) comparing ten loanwords and their equivalents to evaluate their potential to form MWEs.

The rest of this paper is organized as follows. Section 2 presents information on work related to association measures used for MWEs discovery. Methodology is presented in Section 3. It describes the corpus used in this study, lemmas selected as initial seeds, and the four association measures. The results and their evaluation are addressed in Section 4. Finally, the conclusion and plans for future work are presented in Section 5.

2 Related Work

The assumption that MWEs stand out, i.e., they exhibit some sort of salience, allows to extract (or discover) them automatically from texts. This salience is also the reason why especially statistical measures have been so popular when it comes to the discovery of multiword expressions.

Many studies indicate that words that tend to co-occur more frequently than by a pure chance are good candidates for MWEs and propose detecting this statistical significance by measuring the association strength between these words (Manning and Schütze, 1999; Pecina and Schlesinger, 2006; Constantin et al., 2017). Such statistical metrics that can estimate the relationship strength between words in a corpus, based on these words’ co-occurrence count and their individual word counts, are known as association measures (AMs). Since MWEs are characterized by strong collocational behavior, statistical association measures have been widely used in MWEs discovery. The number of proposed association measures over the years has been impressive. More than 30 AMs were described by Evert (2005), Pecina (2008) presented a list with over 80 and new measures as well as their variants are constantly being proposed (Evert, 2008a). However, although numerous studies propose and experiment with association measures performance, there is no consensus on which metric is best for extracting MWEs. Evert (2008a) mentioned that although some measures are more popular and have become standards (e.g., pointwise mutual information, log-likelihood, or t-score), the choice of a suitable metric depends on the particular task as each measure focuses on a different aspect of collocation strength. Since different measures capture various aspects of MWEs, Pecina and Schlesinger (2006) proposed combining some of them and showed that when in combination, AMs can generate better results for MWE discovery than if used in isolation.

The most widely used association measure for MWE discovery is the pointwise mutual information (PMI) proposed by Church and Hanks (1990) for terminology discovery. It is derived for bigrams directly from the mutual information between two random variables, using the log-ratio between the observed co-occurrences of the sequence and the individual words to determine how much the co-occurrence is due to mutual preference. The reported issue with PMI is that it is biased towards infrequent events (Ramisch and Villavicencio, 2018; Villavicencio and Idiart, 2019). Therefore, as observed by Bouma (2009) a moderately associated low-frequency bigram might obtain a better score than a highly associated high-frequency bigram.

Another popular group of AMs used for MWE discovery is based on hypothesis testing. Assuming the null hypothesis that words are independent, their observed and expected counts should be the same. Large values indicate that the candidate words are not independent and can potentially form a MWE. Examples of hypothesis-based AMs are t-score and z-score. They are both based on the assumption of normal distribution, and they work
well for frequent events. However, their usage is not recommended for low-frequency pairs. The z-score test is also not suited for small corpora (Seretan, 2008).

AMs based on contingency tables record the marginal frequencies of the words in an n-gram and the probability of their non-co-occurrence. One such measure is Pearson’s chi-squared test ($\chi^2$) which overcomes the normal distribution problem as it makes no data assumptions. However, $\chi^2$ is again not recommended for small corpora (Manning and Schütze, 1999), and it also tends to prefer common events (Kilgarriff, 1996). Another example is log-likelihood ratio (Dunning, 1993) - a well-known association measure for collocation extraction. It performs well with both frequent and rare events as well as different corpora sizes (Dunning, 1993). However, its reliability is affected by low values of expected frequencies in the contingency table (Pedersen, 1996).

Although AMs have a long history and their utility have been sometimes questioned (e.g., Stubbs, 2002), they are still successfully used in extraction systems, e.g., Evert et al. (2017), Uhrig et al. (2018), Garcia et al. (2019). They also remain an important part of other approaches to MWEs discovery, e.g., Squillante (2014), Tsvetkov and Winter (2014) or Farahmand and Henderson (2016).

3 Multiword Expressions Discovery Methods

3.1 Definition

The definition adopted in this paper is the one presented by Baldwin and Kim (2010) (following Sag et al., 2002): “Multiword expressions (MWEs) are lexical items that: a) can be decomposed into multiple lexemes and b) display lexical, syntactic, pragmatic and/or statistical idiomaticity.” It is one of the most frequently used definitions of MWEs, and it describes the phenomenon this paper focuses on, i.e., multiword constructions displaying some sort of idiomaticity.

3.2 Corpus

The corpus used in the study was sampled from MirasText (Sabeti et al., 2018) corpus - an automatically generated text corpus for Persian. It is one of the largest available Persian corpora, containing 2.8 million documents and over 1.4 billion tokens. The corpus size is 15GB. Each data point is provided with the following information: content, title, content summary and keywords, base website, and exact URL of the webpage.

The content of the MirasText corpus was generated from 250 web pages selected from a wide range of fields to ensure the diversity of data, e.g., news, economy, technology, sport, entertainment, or science.

Since the corpus data was obtained via crawling, it seemed necessary to perform certain cleaning and normalization tasks. Articles containing clipped content were excluded from the final corpus used in this study. The whole corpus data was normalized with Parsivar (Mohtaj et al., 2018) - a tool for processing the Persian language. These steps led to obtaining the final corpus of 50 million tokens, which was used to discover multiword expressions.

3.3 Lemmas

In order to discover Persian multiword expressions with loanwords and their equivalents proposed by the Academy of Persian Language and Literature, a list of 10 pairs (loanword-equivalent pair) was prepared. There were two conditions for choosing these particular lemmas:

1. The Persian lemma is officially proposed as an alternative to the loanword by the Academy of Persian Language and Literature.
2. Lemmas should be part of everyday language, thus belong to general discourse.

Table 1 presents all 20 lemmas (both loanwords and their Persian equivalents) that served as initial seeds to discover MWEs. This table contains the following information: 1) meaning of a lemma, 2) its type, 3) information about lemma’s ambiguity, e.g., lemma ماشین (mâ신) apart from machine, can also mean engine or motor, 4) information about other possible spelling variations of a lemma, e.g.,

---

3The motivation behind targeting MWEs with loanwords lies in the language policy in Iran, which actively proposed native Persian equivalents for borrowed elements. For more details on Iranian language policy see, e.g., Marszalek-Kowalewska (2011) or Moghaddam and Moezzipour (2017).

4Information about other possible meanings come from the following dictionaries:

- online dictionary including a number of Persian monolingual dictionaries (https://www.vajehyab.com)
- online dictionary and thesaurus Abadis (https://dictionary.abadis.ir)
- online Persian glossary based on dictionary of Dehkhoda (https://www.parsi.wiki)
3.4 Association Measures

In order to extract MWE candidates with loanwords and their equivalents, statistical association measures were used. For every lemma, its bi-grams and tri-grams were extracted from 50 million token corpus using the following association methods:

- PMI
- log-likelihood
- t-score
- $\chi^2$ test

These particular AMs were chosen as they are the most popular ones used for the discovery of MWEs (Evert, 2008a; Seretan, 2008; Wahl and Gries, 2018; Villavicencio and Idiart, 2019).

For each association measure, its top 100 bi- and tri-grams per lemma were extracted. This resulted in 1487 unique MWE candidates.

4 Results

4.1 Candidates Filtering

Since association measures produce ranked lists of MWE candidates, their evaluation is usually done through gold standard corpus or manual validation by trained experts.

The outcome of employing AMs to discover Persian MWEs with loanwords and equivalents is a list with 1487 unique MWE candidates. In order to evaluate individual AM performance, all candidates were assessed by external annotators from a crowdsourcing platform. These annotators were linguistically trained native speakers of Persian. The total number of workers contributing to this project was 18, and the inter-annotator agreement (IAA) was calculated with Fleiss’ Kappa - a statistical measure used to evaluate the agreement between three or more raters (Fleiss, 1971).

To ensure the highest quality of annotators’ work, the main part of MWE candidates filtering task was preceded by a trial run on a small gold test set. The IAA on this gold test set was 82%. All contributors’ performance on the gold test set was taken into account, and only annotators with the best performance were invited to perform the main task. Therefore, the final IAA was 87% which indicate that almost perfect agreement was achieved. Annotators were provided detailed guidelines, which included an operational definition of MWEs (as presented in 3.1) and several examples showing true and false MWEs. Each MWE candidate was evaluated by at least three annotators who answered the question: Is the following sequence a valid multiword expression? Possible answers include: YES, NO, and UNABLE TO DETERMINE.

4.2 Association Measures Evaluation

One of the two objectives of this study was to apply and evaluate association measures used to discover MWEs for ten loanwords in Persian and their Persian equivalents proposed by the Academy of Persian Language and Literature. The outcome of applying AMs is a list of MWE candidates. Out of 1487 MWE candidates, 389 turned out to be true MWEs. Figure 1 shows the performance of the four selected association measures when it comes to the discovery of true MWEs.

Figure 1: A number of true MWEs extracted via particular association measures.

As can be seen, the highest number of MWEs were extracted with t-score (248 MWEs), followed closely by log-likelihood (220). Surprisingly, the popular PMI method obtained the worst results, extracting only 148 true MWEs.

In order to further evaluate AMs, precision and recall were computed for all $n$ candidates and plotted as a precision-recall curve. The precision-recall
Finally, figure 4 shows the ratio of MWEs extracted with loanword and equivalent according to selected AMs. In all cases, MWEs with loanwords constitute a bigger group, with best results achieved by t-score (57%), followed by \( \chi^2 \) and log-likelihood (both 54%). The best results for MWEs with equivalent were achieved by PMI (47%). Shared MWEs, i.e., MWEs that occur with both loanwords and equivalents, constitute almost 16% of all true MWEs (when counting shared MWEs only once).

The analysis of MWE candidates rejected by annotators revealed that sequences not labeled as true MWEs tend to belong to one of the following groups: 1) expressions with comparative adjectives, e.g., ماسین ارزان تر cheaper car, 2) expressions with adjectives describing nationalities, e.g., گذرنامه ایرانی Iranian passport, 3) expressions with intensifying adjectives, such as super sport or 4) expressions containing...
4.3 Loanwords and Equivalents Evaluation

All analyzed loanwords turned out to form MWEs, while for 3 of the proposed equivalents (PANORAMA, BRAND, and SYMPOSIUM), no MWEs were found in the present data. The average number of MWEs per loanword is 21 and 17 per equivalent.

Interestingly, the more detailed analysis of extracted MWEs shows that in many cases, loanwords were not only not replaced by the equivalents proposed by the Academy, but the two (loanword and its equivalent) evolved to form distinct MWEs or even MWEs clusters. Pairs that, apart from sharing a substantial number of MWEs, have separate MWEs are: COMPUTER (33% shared MWEs), ONLINE (17% shared MWEs), TECHNOLOGY (18% shared MWEs) and PASSPORT (24% shared MWEs).

Both loanword and equivalent of lemma ONLINE share a substantial number of expressions. Many of the shared MWEs tend to center around the topic of trading, e.g., online trading system, online trader or online stock trading. Analysing all discovered MWEs, it can be observed that the shopping-related theme is quite predominant. Here again, apart from common MWEs (online sale, online payment and online shopping), loanword and its equivalent evolved to have their own MWEs: online purchase, online transaction and online bill in case of equivalent and more place-where-you...

8To check semantic networks for other pairs, please see Appendix B.
can-buy MWEs with loanword: *online shop*, *online store* and *online retail*. Loanword on its own has more negatively associated MWEs, e.g., *online harrassment*, *online attack* as well as expressions referring to gambling, e.g., *online gambling*, *online hazard* and *online casino*. Interestingly, both lemmas form MWE *online encyclopedia* but with two different Persian words for *encyclopedia*.

Loanword and its Persian equivalent TECHNOLOGY share a substantial number of MWEs, most of which represent different technology types, e.g., *information technology*, *face recognition technology* or *nano-technology*. Apart from common MWEs, both lemmas have specialized in certain types, i.e. loanword occurs in the following combinations: *infrared technology*, *quantum technology* or *LTE technology* whereas an equivalent can be found as part of *AI technology*, *HDR technology* and *Bluetooth technology*. What is more, the more positively associated MWEs are the ones with loanword, e.g., *technology upgrade*, *advances in technology* or *technology enthusiast*. The one negatively associated MWE - i.e., *outdated technology* - occurs with the equivalent.

Lemmas expressing PASSPORT differ in the number of MWEs: there are twice as many expressions with equivalent than with loanword. Main topics that can be distinguished here: different passport types, passport parts, passport-related activities and authentication. When it comes to types, there are common MWEs, e.g., *diplomatic passport* or *political passport* and MWEs with equivalent only, e.g., *biometric passport* and *electronic passport*. Passport parts apart from one common: *passport number*, form MWEs with loanword, e.g., *passport photo* and *passport cover*. The activity-related MWEs, except one shared (*issuance of passport*), are all formed with equivalent, e.g., *passport annulment*, *passport renewal* or *passport confiscation*. Finally, there is a cluster of MWEs related to passport authentication, e.g., *passport validity*, *fake passport* and *counterfeit passport*.

In case of MWEs with lemma SPORT, loanword refers more to sporty appearance (i.e. casual yet attractively stylish), e.g., *sporty look*, *sporty model* or *sporty design*. The meaning of sport as a physical activity is employed by MWEs with equivalent, e.g., *sport federation*, *to exercise sport*, *sport activity* or *professional sport*.

In the case of lemma ECOLOGY, there are no shared MWEs. In fact, for the Persian equivalent, only one MWE was found in the corpus, i.e., *ecological economics*.

For lemma MACHINE, there is only one MWE that both loanword and equivalent share: *smart machine*. The loanword tends to form constructions referring to different types of machines, e.g., *washing machine*, *centrifugal machine* and *dishwasher (machine)*. Similar MWEs (also referring to machine types) are found with the Persian equivalent, e.g., *X-ray machine* or *coffee machine*. Since the Persian equivalent is ambiguous, it occurs also in expressions referring to body systems, e.g., *immune system*, *digestive system* and *respiratory system*.

Only MWEs with loanwords were found for the remaining three pairs: PANORAMA, BRAND, and SYMPOSUM. This might be related to a quite late introduction of the equivalent by the Academy (in the case of BRAND and PANORAMA) and to a relatively low raw frequency in the corpus.

## 5 Conclusion

In this paper, an approach to the discovery of Persian MWEs was presented. We focused on a particular group of MWEs: constructions including loanwords in Persian and their native equivalents proposed by the official Iranian body responsible for language reforms - the Academy of Persian Language and Literature. The extraction of MWEs was performed with the use of four popular association measures. There were two goals of this study: 1) to evaluate the performance of association measures for the discovery of MWEs in Persian, and 2) to compare and analyze MWEs with loanwords and MWEs with equivalents.

The former goal was achieved for the four most popular association measures, with t-score performing best with loanword MWEs and PMI with equivalent ones. To our knowledge, it is the first time such analysis was carried out to discover Persian MWEs. The evaluation of MWEs with loanwords and their Persian equivalents was performed for ten pairs, providing information on shared MWEs as well as distinct ones. The complete list of extracted MWEs will be available for translators and students of the Persian language.

Future work includes exploiting a bigger number of association measures and other approaches to MWEs discovery. Moreover, we would like to investigate the impact of genres and context on forming distinct MWEs with loanwords and equivalents.
Acknowledgments

The author would like to thank the anonymous reviewers for their encouraging feedback and insights.

References


A Association Measures

Materials in this appendix section present information about association measures used and compared in this paper (as presented by (Evert, 2008b) and (Evert et al., 2017)).

<table>
<thead>
<tr>
<th>association measure</th>
<th>formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMI</td>
<td>$\log_2 \frac{O_{11}}{E_{11}}$</td>
</tr>
<tr>
<td>log-likelihood</td>
<td>$2 \sum_{ij} O_{ij} \log \frac{O_{ij}}{E_{ij}}$</td>
</tr>
<tr>
<td>t-score</td>
<td>$\frac{O_{11} - E_{11}}{\sqrt{E_{11}}}$</td>
</tr>
<tr>
<td>$\chi^2$ test</td>
<td>$\frac{N((O_{11}O_{22} - O_{12}O_{21}) - \frac{N}{2})^2}{R_1R_2C_1C_2}$</td>
</tr>
</tbody>
</table>

Table 4: Association measures compared in the study.

$O_{ij}$ = contingency table of observed frequencies

$O_{11}$ = observed co-occurrence frequency

$E_{ij}$ = contingency table of expected frequencies

$E_{11}$ = expected co-occurrence frequency

$R_i$ = row sums of the contingency table

$R_1$ = marginal frequency of node

$C_j$ = column sums of the contingency table

$C_1$ = marginal frequency of collocate

$N$ = sample size

B Semantic networks

Table 2: Contingency table for MWE candidate pair: expected values.

<table>
<thead>
<tr>
<th></th>
<th>MWE</th>
<th>¬ MWE</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>$E_{11} = \frac{R_1C_1}{N}$</td>
<td>$E_{12} = \frac{R_1C_2}{N}$</td>
</tr>
<tr>
<td>¬ node</td>
<td>$E_{21} = \frac{R_2C_1}{N}$</td>
<td>$E_{22} = \frac{R_2C_2}{N}$</td>
</tr>
</tbody>
</table>

Table 3: Contingency table for MWE candidate pair: observed values.
The Impact of Text Normalization on Multiword Expressions Discovery in Persian

Katarzyna Marszalek-Kowalewska
YUKKA Lab AG / Berlin, Germany
k.marszalek.kowalewska@gmail.com

Abstract

This paper evaluates normalization procedures of Persian text for a downstream NLP task - multiword expressions (MWEs) discovery. We discuss the challenges the Persian language poses for NLP and evaluate open-source tools that try to address these difficulties. The best-performing tool is later used in the main task - MWEs discovery. In order to discover MWEs, we use association measures and a subpart of the MirasText corpus. The results show that an F-score is 26% higher in the case of normalized input data.

1 Introduction

The field of computational linguistics (CL) and its engineering domain of natural language processing (NLP) has exploded in recent years. It seems to continue to gain momentum because of a straightforward reason: human civilization is drowning in data. In 2008, Google reported that the Web had one trillion pages. In 2016, the number was estimated to 130 trillions. International Data Corporation projects that by 2025, available data may expand to 175 zettabytes. Although these estimates include video, image data, and databases, most of it is plain old text. Unstructured data (also known as free-form text) comprises 70% - 80% of the data available on computer networks. The information content of this resource is unavailable to authorities, businesses, and individuals unless humans read these texts or devise some other means to derive information value from them. And this is where Natural Language Processing comes to the game. NLP procedures can be applied to characterize, interpret, or understand the information content of a free-form text, in other words, to unlock the potential of unstructured data.

However, since the quality of the input data influences the quality of the output, in most cases before the NLP pipeline uses it, this unstructured data needs to undergo certain cleaning and normalization tasks, e.g., removal of extra whitespace, substitution of acronyms, transformation of numerical information, accent removal, substitution of special characters and emoji, or normalization of date format.

This paper addresses the problem of normalizing texts in the Persian language, which is 5th content language for the Web according to W3Tech. In particular, we will focus on the impact of Persian text normalization on one of the downstream NLP tasks - multiword expressions discovery. MWEs are very frequent in language and it has been proved that their proper treatment can make a significant impact on a number of other NLP tasks, e.g. lexicography (Church and Hanks, 1990; Fellbaum, 2016), word sense disambiguation (Finlayson and Kulkarni, 2011), part-of-speech tagging and parsing (Baldwin et al., 2004), information retrieval (Newman et al., 2012), language learning (Christiansen and Arnon, 2017), machine translation (Carpuat and Diab, 2010) or sentiment analysis (Berend, 2011; Williams et al., 2015). To the best of our knowledge, there have been no previous attempts to analyze the text normalization impact on the discovery of MWEs in the Persian language.

The following section provides a brief overview of previous work on text normalization problem. Then, the specific challenges that the Persian language poses for the NLP tasks are described. Section 4 focuses on the impact normalization has
on the discovery of multiword expressions in Persian. It first presents a comparison of different normalization tools and then evaluates the impact of normalizing input data on multiword expressions discovery task.

2 Related Work

Text normalization focuses on transforming noisy (non-standard, informal) text to a more standard representation. Linguistic resources, especially online ones containing slang expressions, acronyms, abbreviations, hashtags, or spelling errors, can deviate a lot from the standard language. Text normalization procedures are applied in order to facilitate NLP applications while dealing with such noisy input.

One of the first studies to indicate the importance of text normalization was done by Sproat et al. (2001) who tried to develop a general normalization process applicable to diverse domains. Since then, the impact of normalizing noisy text and its influence on downstream NLP tasks has been analyzed in a few studies. Han et al. (2013) showed the impact of normalizing social media texts on part-of-speech-tagging. In particular, they focused on tweets and compared original and normalized input texts and different taggers: general Stanford POS tagger and domain-specific Twitter POS tagger. The influence of normalization on parsing was studied by Zhang et al. (2013) who introduced a normalization framework designed with the possibility of domain adaptation. Hassan and Menezes (2013) proposed domain and language independent system based on unsupervised learning for machine translation.

Since text normalization is, in many cases, a necessary preprocessing step for numerous NLP tasks, there are several normalization steps. However, as noticed by Baldwin and Li (2015), it is essential to remember that different normalization tasks would fit different data and downstream NLP applications. Moreover, normalization systems as “one size fits all” seem to be less precise than the tailored ones.

There have been various tasks that consider text normalization as a crucial preprocessing step, the most popular to be spelling correction (Choudhury et al., 2007), statistical machine translation (Aw et al., 2006; Pennell and Liu, 2011) and speech recognition (Kobus et al., 2008). Some unsupervised studies focused on using probabilistic models (Cook and Stevenson, 2009), normalization dictionaries (Gouws et al., 2011), lexicon-based classifiers (Han and Baldwin, 2011) or word association graphs (Sömez and Özgür, 2014).

Recently, there has also been an interest in applying deep learning for normalization procedures. Baldwin et al. (2015) described several systems taking part in shared tasks of Twitter lexical normalization and named entity recognition, underlining that deep learning systems based on lexicon-augmented conditional random fields (CRFs) achieved the best results. Furthermore, a hybrid neural model, which uses word-based encoder-decoder architecture and a character-level sequence-to-sequence model, was introduced for social media text normalization by Lourentzou et al. (2019). Mansfield et al. (2019) addressed, on the other hand, text normalization problem by directly normalizing full sentences using subword models.

There have also been studies into normalizing less standard or low-resource languages. The impact of normalization was evaluated by Agić et al. (2016) in a study on multilingual projection for parsing low-resource languages. An attempt to normalize dialectal Finnish into the normative standard language was presented by Partanen et al. (2019). Hegazi et al. (2021) studied preprocessing of Arabic text on social media. Research on preprocessing tools, including normalizer for Ainu language, was conducted by Nowakowski et al. (2019). Normalization of six low-resource African languages (Afrikaans, Amharic, Hausa, Igbo, Malagasy, Somali, Swahili, and Zulu) was presented by Zupon et al. (2021).

Research on normalizing Persian text focused mainly on addressing the specific challenges (described in section 3) this language poses for NLP tasks. This resulted in a number of processing tools. In 2010, Shamsfard et al. (2010b) proposed STeP1, which provides tokenization, morphological analysis, part-of-speech tagging, and spell checking. ParsiPardaz toolkit, which, apart from providing the same processing steps as STeP1, also includes additional normalization step, was proposed by Sarabi et al. (2013). The first open-source preprocessing tool - Hazm - was introduced by Hazm (2014). Finally, in 2018 Parsivar, another open-source tool, was presented by Mohtaj et al. (2018). Apart from work on preprocessing tools, research in Persian normalization focused also on classification tree and support vector machine (Moattar et al., 2006), N-gram language model combined
with a rule-based method (Panahandeh and Ghanbari, 2019) or sequence labeling models (Doostmohammadi et al., 2020).

3 Challenges of Persian NLP

Research in Persian NLP faces two significant challenges. The first one arises from the number of resources. Although there has been a significant improvement in the number of available NLP resources e.g. Hamshahri Corpus (Darrudi et al., 2004), Bijankhan Corpus (Bijankhan et al., 2011), FarsName (Hajitabar et al., 2017), ShEMO (Nezami et al., 2019), Persian Dependency Tree-bank (Rasooli et al., 2013), SentiPers (Hosseini et al., 2018), FarsNet (Shamsfard et al., 2010a) or PersBERT (Farahani et al., 2020) in recent years, Persian is still a heavily unresourced language compared to English or German. The second problem is related to the challenging character of Persian itself and inconsistencies in the writing system. The following section discusses the main challenges the Persian language poses to NLP applications.

3.1 Encoding

One of the first problems in processing Persian texts is the existence of different character encodings. While creating digital texts, both Persian Unicode characters and Arabic ones are sometimes used. As a result, for example the letter ی [ye] can be expressed by 3 different encodings: either the Persian one: \u06a9, or two Arabic encodings: \u06cc or \u0649 (Sarabi et al., 2013; Ghayoomi and Mommazi, 2009; Megerdoomian, 2018).

3.2 Writing System

The Persian writing system poses several difficulties with regard to NLP. First of all, Persian letters can have joiner and non-joiner forms based on their position in a word. This feature is quite common among languages, yet in Persian certain letters written at the end of a word may not be joined. Some users treat them as separate characters and do not use whitespace after the word. As a result, tokenization is not always reliable.

Moreover, foreign (borrowed) elements in Persian tend to be written arbitrarily, i.e., the fact that there are, for example, four possible forms of letter ‘ز’ (ظرفی) poses certain difficulties for users. Although the Academy of Language and Literature tries to systemize it, there is still great arbitrariness when it comes to actual usage. As an example, consider the following variants of the borrowed word bulldozer in Persian:

• بولدوژر
• بولدوزر
• بولدوذر
• بولدوضر

Furthermore, there are no capital letters, which may cause ambiguity for the named entity recognition task. The lack of capital letters can also cause problems with the identification of acronyms.

Another challenge of the writing system is text directionality. Although letters are written from right to left, numbers are written in the opposite direction, e.g.

ایران ۱.۲ میلیون بشکه نفت خام صادر کرد

Iran exported 1.2 million barrels of crude oil’.

What is more, it is not uncommon for users to use Arabic numerals instead of Persian ones, e.g.

کنفرانس در سال ۱۹۹۷ اتفاق افتاد

The conference took place in 1997’.

3.3 Word and Phrasal Boundaries

In Persian, as in many other languages, whitespace designates the word boundary. However, apart from the standard whitespace, there is also zero-width-non-joiner space (known as pseudospace) used with non-joiner letter forms. In fact, the whitespace and pseudospace are used inconsistently, causing tokenization and segmentation sometimes really challenging.

As mentioned in 3.2, Persian letters have different forms depending on their position in a word. Thus, users often treat non-joiner forms incorrectly, i.e., not adding whitespace after them, e.g.,

تو یا گرفت ‘YouForgotFromUs’. As a result, this phrase would be processed as one lexeme instead of four separate ones, i.e.,

تو از ما گرفت.

On the other hand, whitespace is often used instead of pseudospace which causes words such as

‘linguistics’ to be processed as two separate words

‘شناس’ ‘knowledge’

is the official Iranian regulatory body of the Persian language.
(when written with whitespace, i.e., /uni06CC.fina/afii57427.init/afii57415.fina/afii57446.medi/afii57428.init ... the MWE lexicon. To our
knowledge, there have not been any studies that
address the discovery of MWEs in Persian with
written forms but with different pronunciations,
As a result, word and phrase boundaries are of-
(determination. In most cases, it is pronounced but
directly related to complex lexemes, consisting of
and attached affixes that represent a sep-
artate lexical category or part of speech from the
one they are attached to. A few examples of this
situation are presented in table 1.

3.4 Ambiguity
Dealing with word sense ambiguity is one of the
main NLP challenges. This task is particularly
difficult in the case of Persian as the number of
heterophonic homographs (words with identical
written forms but with different pronunciations,
each associated with a different meaning) is high.
The main reason for this situation is the fact that
Persian short vowels are usually not written. There-

fore, the word ملک could be interpreted in the four
following ways:

• ملک [malak] ‘angel’,
• ملک [malek] ‘prince’,
• ملک [melk] ‘domain’,
• ملک [molk] ‘country, territory’.

3.5 Ezafe Construction
Ezafe is a syntactic construction used to express
determination. In most cases, it is pronounced but
not written (since it is expressed by a short vowel),
contributing to ambiguity, especially in chunking
and semantic as well as syntactic processing of a
sentence. Hence, the following sentence can be
interpreted in two different ways depending on the
presence of ezafe:

پدر حسن را دید
1. [pedar hasan ra did] ‘Father saw Hassan.’
2. [pedare-e hasand ra did] ‘He/She saw Has-
san’s father.’

4 Normalization Impact on Multiword
Expressions Discovery
The challenges presented above: inconsistency in
using white- and pseudospace, different encodings,
missing short vowels or bidirectionality can pose
many difficulties for proper processing of Persian
for several NLP tasks. Therefore, a certain level
of text normalization seems necessary. The fol-
lowing section describes the impact normalization
procedures have on the discovery of multiword
expressions task.

4.1 Multiword Expressions Discovery
Linguistics expressions that consist of at least two
words (even when represented by a single token)
and are syntactically and/or semantically idiosyn-
cratic - this is probably the most common definition
of multiword expressions. They attracted a lot of
research attention and have been the main topic in
plenty of papers.

MWEs are very frequent in language and range
over a number of different linguistic constructions,
from idioms, e.g. to kick the bucket, to fixed expres-
sions, e.g. fish and chips, light verb constructions,
e.g. give a demo, to noun compounds, e.g. traffic
light. Biber et al. (1999) claim that the number of
MWEs in spoken English is 30% – 45% and 21%
in academic prose. Jackendoff (1997) suggests that
the number of MWEs in a speaker’s lexicon is the
same as simple words, yet if we take into consid-
eration the domain-specific lexicons, this number
seems to be an underestimation (Sag et al., 2002).
Indeed, the research conducted by Ramisch (2009)
suggests that the MWEs ratio can be between 50%
and 80% in a corpus of scientific biomedical ab-
estimate that MWEs can constitute more than 70%
of the specialized lexicon.

MWEs processing consists of two tasks: identi-
fication and discovery (Constant et al., 2017).
MWEs identification focuses on tagging a corpus
with actual MWEs. The research on MWEs in Per-
sian has so far focused mainly on the identifica-
tion of verbal multiword units and light verb con-
structions (LVCs) in particular, e.g., Taslimipoor et al.
(2012); Salehi et al. (2012, 2016). MWE discovery -
the task this paper tries to address - is a process
that focuses on finding new MWEs (types) in cor-
pora and storing them, e.g., in the form of a lexicon,
for further usage. This task takes text as input and
generates a list of MWE candidates from it. These
candidates can be further filtered and evaluated
by trained experts. True MWEs are stored in a
repository or added to the MWE lexicon. To our
knowledge, there have not been any studies that
address the discovery of MWEs in Persian with

936
respect to the normalization of input text.

The assumption that MWEs stand out, i.e., they exhibit some salience, allows us to extract (or discover) them automatically from texts. This salience is also why especially statistical procedures, such as association measures (AMs), have been so popular when it comes to MWEs discovery. This paper also approaches the discovery of MWEs by employing a selected set of association measures.

4.2 Corpus

The corpus used in the study was MirasText (Sabeti et al., 2018) corpus - an automatically generated text corpus for Persian. It is one of the largest available Persian corpora, containing 2.8 million documents and over 1.4 billion tokens. The corpus size is 15GB. Each data point is provided with the following information: content, title, content summary and keywords, base website, and exact URL of the webpage.

The content of the MirasText corpus was generated from 250 web pages selected from a wide range of fields to ensure the diversity of data, e.g., news, economy, technology, sport, entertainment, or science.

Corpus content was generated through crawling; thus, there is a possibility of including duplicated texts. In order to remove duplicated content from the corpus Sabeti et al. (2018) used a filtering process based on a bloom filter (Almeida et al., 2007).

4.3 Normalization

4.3.1 Processing Tools Evaluation

To ensure that the best normalization tool is used for the discovery of MWEs task, firstly, research comparing two open-source processing tools for Persian was carried out. These tools are Hazm and Parsivar, and they both provide normalization, tokenization, chunking, and part-of-speech steps. In order to evaluate these tools, a small corpus of 5000 sentences was annotated by 3 Persian linguistic experts with respect to sentence segmentation and tokenization. The inter-annotator agreement was calculated with Fleiss’ Kappa - a metric used to evaluate the agreement between three or more raters (Fleiss, 1971) and annotators achieved 98% which indicates almost perfect agreement. 6 Table 2 presents the tokenization results of normalized and raw data.

<table>
<thead>
<tr>
<th>Tool Type</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hazm not-normalized</td>
<td>71%</td>
<td>72%</td>
<td>72%</td>
</tr>
<tr>
<td>Hazm normalized</td>
<td>97.5%</td>
<td>97%</td>
<td>97%</td>
</tr>
<tr>
<td>Parsivar not-normalized</td>
<td>79%</td>
<td>75%</td>
<td>77%</td>
</tr>
<tr>
<td>Parsivar normalized</td>
<td>99%</td>
<td>98%</td>
<td>98%</td>
</tr>
</tbody>
</table>

Table 2: Tokenization results.

The better tokenizer turned out to be Parsivar (Mohtaj et al., 2018) achieving 98% F-score. The superior performance of Parsivar over Hazm was also confirmed in the Persian plagiarism detection study by (Mohtaj et al., 2018). It seems that the main difference between these two tools lies in the better performance of space correction by Parsivar.

Nevertheless, what seems to be of higher importance here is the fact that the results obtained using raw and normalized corpus differ significantly. Regardless of the preprocessing tool used, the tokenizer performance was in both cases more than 20% higher in the case of the normalized data.

4.3.2 Corpus and Its Normalization

Since the MirasText corpus data was obtained via crawling, it seems necessary to perform certain cleaning and normalization tasks. The initial corpus analysis showed that a certain number of articles contain incomplete content (clipped content). Such articles were excluded from the final corpus used in this study. After filtering out the clipped articles, the total number of corpus documents was 2,072,521. As the next step, 50 million token corpus for the discovery of MWEs was sampled.

For most of the NLP tasks, the first necessary step is to tokenize the input text. However, as already mentioned, this is not a simple task in Persian text processing since there are two kinds of spaces:

---

6For interpretation see Landis and Koch (1977).
white- and pseudospace, which are not used consistently. Using inconsistent spacing results in high ambiguity, both on lexical and syntactical levels. Therefore, for a corpus of millions of documents written by thousands of various authors, it is necessary to unify its data, and one of the first and most essential unification steps in Persian NLP is to correct spaces.

As a result of an experiment described in 4.3.1, the best processing tool turned out to be Parsivar (Mohtaj et al., 2018), and the corpus used for the discovery of MWEs task was normalized with it. Parsivar, in its normalization task apart from encodings and numbers unification, performs two different types of space correction:

- rule-based space correction: a set of rules using regular expressions were employed in order to detect spaces within words correctly, e.g., \( /\text{مش روم} /\text{تحلیل} /\text{گور} (\text{miravam}) 'I am going' or \\
\( /\text{تحلیل} /\text{گور} (\text{tahlilgar}) 'analyzer'. The problem with words that consist of two or more tokens but cannot be extracted with one of these rules was addressed by constructing a dictionary. This helped with words as \\
\( /\text{گفت و گو} (\text{goft-e gu}) 'conversation'.

- learning-based space correction: using training model that recognizes multi-token words as one token. Parsivar uses 90% of the Bijankhan corpus (which contains multi-word tokens annotated with IOB tagging format) as training data. Naïve Bayes model was used to find word boundaries. The model was evaluated on the remaining 10% of Bijankhan corpus and got 96.5% of F-score for space correction on that validation set.

Table 3 presents raw and normalized metrics of sentence segmentation and tokenization performed on the corpus used in the present study.

<table>
<thead>
<tr>
<th>Task</th>
<th>Not-normalized</th>
<th>Normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of sentences</td>
<td>1,464,996</td>
<td>1,537,725</td>
</tr>
<tr>
<td>number of tokens</td>
<td>52,536,988</td>
<td>51,525,867</td>
</tr>
</tbody>
</table>

Table 3: Evaluation metrics of the corpus.

As can be seen, both the number for sentence segmentation and tokenization differ significantly (the difference in the number of tokens is almost 1 million!) if we compare the corpus before and after normalization. The difference in sentence segmentation stems from the incorrect treatment of dots in the not-normalized corpus, especially in case of numerals, dates, webpages and in combination with other punctuation marks. These results show that proper cleaning and normalization tasks (especially unifying spaces) are crucial during Persian text processing.

4.4 Methodology

In order to extract Persian multiword expressions, a list of 20 lemmas that would serve as initial seeds was prepared. The task of MWEs discovery was addressed from a statistical perspective. For every lemma, its bi-grams and tri-grams were extracted separately from raw and normalized corpus using the following association methods:

- PMI
- log-likelihood
- t-score
- \( \chi^2 \) test

These particular AMs were chosen as they are the most popular ones used for the discovery of MWEs (Evert, 2008; Seretan, 2008; Wahl and Gries, 2018; Villavicencio and Idiart, 2019).

For each association measure, its top 100 bi- and tri-grams per lemma were extracted - this resulted in 1487 unique MWE candidates from the normalized corpus and 1817 from the raw one.

5 Results

5.1 Candidates Filtering

The outcome of employing association measures to discover Persian MWEs is a list with 1487 unique MWE candidates from the normalized corpus and a list with 1817 unique MWE candidates from the raw corpus. All MWE candidates were evaluated by trained experts - Persian native speakers with linguistic background.

Annotators were provided detailed guidelines which included an operational definition of MWEs (“Multiword expressions (MWEs) are lexical items that: a) can be decomposed into multiple lexemes and b) display lexical, syntactic, pragmatic and/or statistical idiomaticity” as presented by Sag et al., 2002) and a number of examples presenting true and false MWEs. Each MWE candidate was evaluated by at least three annotators who answered the question: Is the following sequence a valid multiword expression? Possible answers include: YES, NO, and UNABLE TO DETERMINE.
The total number of experts contributing to this project was 21, and the inter-annotator agreement (IAA) was calculated again with Fleiss’ Kappa. All annotators were working on both sets: MWE candidates extracted from raw and normalized corpus. The IAA results were 87% and 81% for normalized and raw corpus, respectively. Thus, the final average IAA for this task was 84% which indicate that almost perfect agreement was achieved.

5.2 Multiword Expression Discovery Evaluation

After evaluating the candidates, the number of true MWEs in a normalized corpus was 389 and 154 in the raw one.

The main objective of this study was to evaluate the impact of text normalization on the MWEs discovery task in Persian. Figure 1 shows the performance of the four selected association measures when it comes to the discovery of true MWEs.

![Figure 1: A number of true MWEs extracted with analyzed association measures.](image)

As can be seen, each AM performs better when used with the normalized data. The highest number of MWEs were extracted with t-score (248 MWEs), followed closely by log-likelihood (220 MWEs), both performed on the normalized corpus.

The number of true MWEs is, however, not enough to evaluate the performance. Therefore, it is interesting to perform error analysis and check which cases were and which were not discovered in the raw corpus (compared to the normalized one). Correctly detected MWEs in the raw corpus can be divided into three categories:

- MWEs with Arabic numerals, e.g. 360 panorama,
- MWEs with words written in Latin script, e.g., HDR technology,
- MWEs whose components do not contain non-joiner letters, e.g., supercomputer.

True MWEs discovered in the normalized corpus but not in the raw one seem to have generally one thing in common: they contain words with non-joiner letters; therefore, the use of whitespace is not always consistent. Examples of MWEs discovered in normalized corpus but missed in the raw one are فروش بخش online sales, بازی رایانهای computer game, اکولوژی دریا sport club, or ماشین رهگیری marine ecology. Furthermore, all MWEs found in the raw corpus were also discovered in the normalized one.

In order to further evaluate true MWEs discovered using raw and normalized corpus, we used the combined outcome from all AMs. For MWE candidates from raw and normalized corpus, precision, recall, and F-score were computed (similarly to Evert and Krenn, 2001 who used these metrics to plot a precision-recall curve for direct comparison of different AMs). The overall impact of text normalization on the discovery of multiword expressions in Persian is presented using F-score in table 4.

<table>
<thead>
<tr>
<th></th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not-normalized</td>
<td>15%</td>
</tr>
<tr>
<td>Normalized</td>
<td>41%</td>
</tr>
</tbody>
</table>

Table 4: Comparison of F-score.

The F-score turned out to be 26% higher in the case of normalized data. Therefore, applying text normalization procedures proved to have a significant impact on the discovery of multiword expressions task in Persian.

Since the different normalization steps may vary in the impact on the downstream NLP tasks, their performance for discovering MWEs in Persian was also analyzed. Table 5 presents F-score for all text normalization steps (performed separately as well as in various combinations). It turned out that the most efficient combination of normalization steps is the unification of encodings and dates combined with space correction. In fact, correcting and unifying spaces proved to be the most crucial normalization step for the presented task.

6 Conclusion and Future Works

In this paper, an impact of text normalization on a downstream NLP task was presented. In particular, we focused on the normalization of Persian language data for multiword expression discovery.
The experiment results show that the performance of a system without a Persian-tailored normalization step is 26% worse (F-score), which is a significant deterioration. To our knowledge, this was the first time when the influence of text normalization on the discovery of multiword expressions in Persian was described.

Since this paper focuses on normalization as a preprocessing step, it would be interesting to compare its impact with post-processing tasks. Some further future works include analyzing how normalized data influences other NLP tasks in the Persian language, particularly syntactic parsing and sentiment analysis. Moreover, we would like to compare the tools described in this paper with a neural network approach to text normalization.

Acknowledgments

The author would like to thank the anonymous reviewers for their encouraging feedback and insights.

References


Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Basu Anupam.


Hazm. 2014. Python library for digesting Persian text.


943
Improving Neural Language Processing with Named Entities

Kyoumoto Matsushita
Fujitsu Ltd. / Tokyo, Japan
m.kyoumoto@fujitsu.com

Takuya Makino
Fujitsu Ltd. / Tokyo, Japan
makino.takuya@fujitsu.com

Tomoya Iwakura
Fujitsu Ltd. / Tokyo, Japan
iwakura.tomoya@fujitsu.com

Abstract

Pretraining-based neural network models have demonstrated state-of-the-art (SOTA) performances on natural language processing (NLP) tasks. The most frequently used sentence representation for neural-based NLP methods is a sequence of subwords that is different from the sentence representation of non-neural methods that are created using basic NLP technologies, such as part-of-speech (POS) tagging, named entity (NE) recognition, and parsing. Most neural-based NLP models receive only vectors encoded from a sequence of subwords obtained from an input text. However, basic NLP information, such as POS tags, NEs, parsing results, etc, cannot be obtained explicitly from only the large unlabeled text used in pretraining-based models. This paper explores use of NEs on two Japanese tasks; document classification and headline generation using Transformer-based models, to reveal the effectiveness of basic NLP information. The experimental results with eight basic NEs and approximately 200 extended NEs show that NEs improve accuracy although a large pretrained model trained using 70 GB text data was used.

1 Introduction

In statistical NLP technologies, which were widely employed before neural-based NLP emerged, basic information, such as POS tags and NEs were often used as features for document classification (Higashinaka et al., 2012) and other NLP tasks. However, since neural-based NLP technologies have emerged with state-of-the-art (SOTA) performances, basic NLP technologies, such as POS tagging and NE recognition are no longer used for obtaining features from given text. This is because most neural-based NLP methods attain higher accuracy using only sentence representation encoded by a pretraining model learned from the large scale unlabeled text.

However, we think there are some rooms that basic information, such as POS tags and NEs contribute to recent neural-based NLP even if large scale pretrained models such as BERT (Devlin et al., 2019) and BART (Lewis et al., 2020) are used for obtaining sentence representations.

One of the reasons is that POS tags and NEs are usually obtained from outputs of analyzers trained from labeled training data created by human-beings for each specific purpose. For example, NE recognizers can identify single words or phrases with their classes, such as PERSON, ORGANIZATION, and so on, which are not explicitly given from pretrained models. Therefore, we think different kinds of information compared with pretraining ones are obtained from outputs of such NLP tools and we expect such information contributes to further improve accuracy.

We propose the incorporation of basic NLP information to neural NLP architectures and evaluate their effectiveness on two Japanese NLP tasks. In this paper, two NE categories, eight basic NEs (Sekine and Isahara, 2000) and approximately 200 extended NEs (Sekine and Nobata, 2004), are considered. By combining NE information with a pretrained model, we train a model of each task.

Experimental results on document classification using the BERT model and headline generation using the BART model, show that combining NEs with a large pretrained model contributes to significantly improved accuracy.

2 Experimental Design

This paper investigates the effectiveness of NE information for SOTA NLP models. In order to investigate effectiveness of NEs for SOTA NLP technologies, we conducted experiments on the two tasks. The first one is document classification for
investigating the effectiveness on a classification task. The one is headline generation for investigating the effectiveness on a generation task. We use BERT (Devlin et al., 2019) for document classification and BART (Lewis et al., 2020) for headline generation as pretraining models.

To evaluate these tasks, we defined an architecture that uses NE class embeddings in addition to subword embeddings given by one of the pretrained models, for each task. We refer to the architectures as BERT\textsuperscript{NE} and BART\textsuperscript{NE}, where NE indicates the NE class definition. For the inputs of BERT\textsuperscript{NE} and BART\textsuperscript{NE}, we recognize NEs in texts with an NE recognizer. Finally, using BERT\textsuperscript{NE} and BART\textsuperscript{NE}, we evaluated the models with and without NE annotations on the two tasks for examining the effectiveness of NE annotations in each task.

One expected effect is the impact of granularity of NE classes on the accuracy of each task. To investigate such an effect, we use the following Japanese NE categories, described in Section 5.

- Basic Named Entity (BNE): eight types the basic NEs defined by the IREX committee (Sekine and Isahara, 2000).
- Extended Named Entity (ENE): approximately 200 types of ENE classes (Sekine and Nobata, 2004).

Furthermore, to evaluate the impact of NE recognition accuracy, we use the following two NE recognition methods.

- FNER: a feature-based NE recognizer (NER) (Iwakura, 2011)
- NNER: a neural-based NER (Akbik et al., 2018)

For FNER and NNER, we trained models of the two NE class definitions for each NE recognizer. Four NE recognizers were used in this experiment. The accuracy of NNER exceeds that of FNER. In our internal evaluation with the IREX GENERAL data for the BNE definition, the accuracy of NNER for BNE is 93.44, which is 2.07 points higher F-measure than 91.37 of FNER. With these two NE recognizes, we investigate the impacts of NER accuracy on the performance of document classification and headline generation.

We use the following terms for the different settings.

\[
\text{PRETRAIN}_{\text{NER}}, \text{NEC}_{\text{NER}}, \text{PRETRAIN}_{\text{FNER}}, \text{NNER}_{\text{FNER}}
\]

where \text{PRETRAIN} is BERT or BART, \text{NEC} is one of BNE and ENE, and \text{NNER} is one of FNER and NNER. For example, BERT\textsuperscript{NEC}_{\text{FNER}} indicates BERT with ENE classes gives from FNER.

\section{Baseline Models}

Here, we use two pretraining architectures, a pretrained BERT model for document classification and a pretrained BART model for headline generation.

\subsection{BERT-based Document Classification}

BERT is a Transformer-based pretraining architecture showing high performances in various NLP tasks. It generates encoded embeddings of each input token using a Transformer-based bidirectional encoder. Then, a target task architecture uses the embeddings.

For the document classification task, document classifiers use \(h_{[CLS]}\), encoded embedding of [CLS] token by BERT, and classify the document based on \(h_{[CLS]}\). This is because \(h_{[CLS]}\) is used as the aggregate sequence representation for document classification task in BERT architecture.

\[
o_{[CLS]} = \text{classifier}(h_{[CLS]}),
\]

where \([CLS]\) is a special token corresponding to the beginning of input tokens; \(h_{[CLS]} \in \mathbb{R}^D\) is an embedding of [CLS] token encoded with BERT; \(D\) is the number of dimensions of the encoded embeddings, \(o_{[CLS]} \in \mathbb{R}^C\) is a score vector of classes and \(C\) is a number of document classes. We employ \(\arg\max(o_{[CLS]})\) as a predicted document class.

The embeddings of the [CLS] of each document is obtained with up to first 510 tokens in an input and [SEP] token that is also one of special tokens indicating the end of an input. The embeddings of each token of BERT is represented as

\[
e_{\text{input}} = e_{\text{pos}} + e_{\text{ttype}} + e_{\text{token}}, \quad (1)
\]

where \(e_{\text{pos}}\) is relative position embeddings for relative position \(\text{pos}\) of a token from the [CLS]; \(e_{\text{ttype}}\) is token type embeddings for \(\text{ttype}\) indicating id (such as a natural number) of each sentence included in an input; \(e_{\text{token}}\) is token embeddings for a token \(\text{token}\). Figure 1 shows an example of generating input embeddings from input tokens.

\subsection{BART-based Headline Generation}

BART is also a Transformer-based pretraining architecture, especially for sequence-to-sequence
Figure 1: Example of generation of input embeddings of BERT.

<table>
<thead>
<tr>
<th>Input Tokens</th>
<th>[CLS]</th>
<th>Alan</th>
<th>Turing</th>
<th>educated</th>
<th>at</th>
<th>King’s</th>
<th>College</th>
<th>[SEP]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token Embeddings</td>
<td>$e_{[CLS]}$</td>
<td>$e_{Alan}$</td>
<td>$e_{Turing}$</td>
<td>$e_{educated}$</td>
<td>$e_{at}$</td>
<td>$e_{King’s}$</td>
<td>$e_{College}$</td>
<td>$e_{[SEP]}$</td>
</tr>
<tr>
<td>Token Type Embeddings</td>
<td>$e_A$</td>
<td>$e_A$</td>
<td>$e_A$</td>
<td>$e_A$</td>
<td>$e_A$</td>
<td>$e_A$</td>
<td>$e_A$</td>
<td>$e_A$</td>
</tr>
<tr>
<td>Position Embeddings</td>
<td>$e_0$</td>
<td>$e_1$</td>
<td>$e_2$</td>
<td>$e_3$</td>
<td>$e_4$</td>
<td>$e_5$</td>
<td>$e_6$</td>
<td>$e_7$</td>
</tr>
</tbody>
</table>

Table: Example of generation of input embeddings of BERT.

This architecture shows high performance sequence-to-sequence tasks, such as headline generation and document summarization. BART, which consists of only an encoder. The pretraining follows an autoencoder approach in which an input sentence is generated from the input sentence. For example, “A B C” is an input, BART uses a variant of the sentence, in which words are shuffled like “B A C” or a masked sentence like “A _ C” as inputs for generating “A B C”.

BART encoder generates encoded contextual embeddings of subwords, $\mathbf{H}_{\text{enc}} \in \mathbb{R}^{D \times N}$, from an input subword sequence with a bidirectional encoder as in BERT, where $D$ is a dimension of contextual embeddings and $N$ is the length of an input sentence. Eq. (1) is used of each subword for the BART encoder.

$$t_i = \text{BART}_{\text{DEC}}(t_{\text{dec},i-1}, \mathbf{H}_{\text{enc}}),$$

where $t_i$ is the $i$-th token of the decoded tokens; $t_{\text{dec},i}$ is $i$ decoded tokens (i.e., $t_{\text{dec},i} = (t_0, t_1, \ldots, t_{i-1})$). The $t_0$ is a special token representing the head of the decoded tokens and is denoted by “<s>”.

### 4 Named Entity-based Models

This section describes a BERT-based document classification and BART-based headline generation using NE information.

#### 4.1 Use of NE Information

The difference for both base models is the representation of $e_{\text{input}}$ defined by Eq. (1).

To incorporate NE information into BERT and the encoder of BART, we use $e_{\text{input}}$ defined as

$$e_{\text{input}} = e_{\text{pos}} + e_{\text{NE}} + e_{\text{token}},$$

where $e_{\text{NE}}$ corresponding to each NE class is use, instead of $e_{\text{type}}$.

Figure 2 shows an example of input embeddings from input tokens with NE classes.

#### 4.2 Parameter Updates for NE-based Models

Since we used $e_{\text{NE}}$ instead of $e_{\text{type}}$, accuracy may be degraded by using the same fixed pretrained parameters of BERT/BART. To avoid this degradation, we finetune all parameters of BERT/BART using the training data of the target task.

Another option is training BERT and BART models with NE classes from scratch. However, we chose to finetune the original pretrained models that do not have $e_{\text{NE}}$ for the following reasons:

- Our method takes advantage of existing pretraining models to easily incorporate different definitions of NEs since it does not require any pretraining to incorporate NEs.
- The maximum NE class types are approximately 200 of the ENE definition. Therefore, the finetuning approach can adopt NE embeddings based inputs.
- For the BERT, the use of a publicly available model is a fair comparison because the model was trained not for our purpose.

The proposed method can be applied to other huge pretraining methods, such as RoBERTa (Liu et al., 2019) for document classification and PEGASUS (Zhang et al., 2020) for headline generation. Even if we cannot obtain the same huge pretraining dataset, we can use existing pretraining models to enhance them with NEs.

1The finetuning of BERT$^{\text{NE}}$ and BART$^{\text{NE}}$ from pretrained BERT and BART excludes $e_{\text{NE}}$.

2Unfortunately, we could not find any models of BART for Japanese. Therefore, we trained a Japanese model for BART.
NE class | Example
--- | ---
ARTIFACT | Nobel Prize in Chemistry
LOCATION | Bulgaria
ORGANIZATION | King’s College
PERSON | John Smith
DATE | May
MONEY | 100 USD
PERCENT | 100%
TIME | 10:00 a.m.

Table 1: NE examples of BNE.

5 Named Entity Definition

This section introduces the two NE definitions used in our experiments, a basic NE category defined at Retrieval and Extraction Exercise (IREX) (Sekine and Isahara, 2000) and Extended Named Entity (ENE) definition (Sekine and Nobata, 2004).

- **Basic Named Entity (BNE):** Table 1 shows an example of the BNE definition. BNE consists of eight NE classes, PERSON, ORGANIZATION, DATE, TIME, LOCATION, PERCENT, MONEY and ARTIFACT, and a special class OPTIONAL. The OPTIONAL is used when annotators cannot uniquely decide the NE class of each NE. In our experiments, we excluded the OPTIONAL NE annotations in the training and evaluation phases.

- **Extended Named Entity (ENE):** The ENE definition has over 200 NE classes associated with a hierarchy. The IGNORE class of ENE represents the excluded parts. The CONCEPT class of ENE represents entities that cannot be classified into other ENE classes. In our experiments, we excluded IGNORE and CONCEPT NE classes in the training and evaluation phases.

Table 2 shows examples of NE class annotations with the two definitions. The ENE definition is more elaborated than that of BNE. For example, the BNE class of “King’s College” is ORGANIZATION, however, that of ENE falls under School, which is a subcategory of ORGANIZATION.

6 Data Sets

This section introduces data sets for document classification and headline generation, created from the articles of Mainichi newspaper.

6.1 Document Classification Data Set

The document classification dataset was created from Mainichi newspaper data (Mai-news). In this dataset, we used 2019 year articles as test data, 2018 year articles as development data, and articles of 2013-2017 years as training data. We used this split setting because, in practical situations, we usually have to train a model to predict future target information using past data. We used a pre-processed main body text of each Mai-news article by a procedure described in the Appendix, as an input text on document classification.

Table 3 shows the document classes. We used 14 categories of news articles as the target classes of our document classification task. The document class of each article is recorded in its AD attribute. We refer to this dataset for document classification as Mai-news-dc. The second column of Table 4 shows the statistics of Mai-news-dc.

6.2 Headline Generation Data Set

The dataset for headline generation was also created from Mai-news. As in the document classi-
### NE category Example of Annotations

<table>
<thead>
<tr>
<th>NE category</th>
<th>Example of Annotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without NE</td>
<td>Alan Turing educated at King’s College.</td>
</tr>
<tr>
<td>BNE</td>
<td>&lt;PERSON&gt;Alan Turing&lt;/PERSON&gt; educated at &lt;ORG&gt;King’s College&lt;/ORG&gt;.</td>
</tr>
<tr>
<td>ENE</td>
<td>&lt;Person&gt;Alan Turing&lt;/Person&gt; educated at &lt;School&gt;King’s College&lt;/School&gt;.</td>
</tr>
</tbody>
</table>

Table 2: Examples of annotation each NE category. “ORG” indicates ORGANIZATION NE class of BNE.

<table>
<thead>
<tr>
<th>Document Class</th>
<th>Commentary</th>
<th>Editorial</th>
<th>World</th>
</tr>
</thead>
<tbody>
<tr>
<td>Economy</td>
<td>Special Topic</td>
<td>Culture</td>
<td></td>
</tr>
<tr>
<td>Household</td>
<td>Sports</td>
<td>Society</td>
<td></td>
</tr>
<tr>
<td>Science</td>
<td>Life</td>
<td>Entertainment</td>
<td></td>
</tr>
<tr>
<td>Multi Discipline</td>
<td>Reading books</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Document classes used in document classification.

7 Experimental Setting

This section shows the pre-process of input texts, hyperparameters, evaluation metrics, and so on.

7.1 Pretrained Models

We used cl-tohoku/bert-base-japanese-whole-word-masking[^6] as our Japanese BERT pretrained model. This model was pretrained with approximately 17 million sentences of Japanese Wikipedia articles. The configuration is the same as the original BERT.

For BART, we used the 70 GB Japanese dataset of CC-100[^7] for pretraining BART. This is because no Japanese BART models were publicly available. The Japanese version of BART is pretrained 178,000 steps with a mini-batch size of 1,024 on 64 NVIDIA Volta 100 GPUs. For tokenizing texts, we used a unigram language model-based subword tokenization (Kudo and Richardson, 2018) trained on the same data used to pretrain the Japanese version of BART.

7.2 Training Models

We fine-tuned BERT[^NE] models using Mai-news-dc annotated by BNE and ENE categories, and BART[^NE] models with Mai-news-hg annotated by the two NE categories only one time. For each evaluation, these models were run once. The Appendix includes other hyperparameters.

7.3 Named Entity Annotation to Text

We annotated texts of Mai-news-dc and Mai-news-hg datasets with NE classes using NE recognizers, a classic feature-based NE recognizer (Iwakura, 2011) and a pretraining-based NE recognizer (Akbik et al., 2018). We refer to the former as FNER and the latter as NNER.

7.3.1 Document Classification

First, tokenization of the input texts was performed. Here, a text is first tokenized by MeCab[^9], a Japanese morphological analyzer that segments words with their POS tags from a text, with IPA dictionary[^10]. Then, cl-tohoku/bert-base-japanese-whole-word-masking’s tokenizer based on the WordPiece (Schuster and Nakajima, 2012) was applied to tokenized text with MeCab to decompose each word into subwords. We en-
enhanced the cl-tohoku/bert-base-japanese-whole-word-masking’s tokenizer with special tokens for NE classes such as “<PERSON>” and “</PERSON>”. Then, each \( e_{\text{NE}} \) was assigned to its corresponding subword and NE class tokens were removed.

After preprocessing, we fine-tuned a model not only targeting model-specific layers but also BERT layers.\(^{11}\)

The model was evaluated using development data every 1,000 batch steps in a training phase and if the model achieves the best accuracy on the development data, we kept the model. The final accuracy of the experiments was calculated using the kept model.

7.3.2 Headline Generation

We tokenized the input source and target texts using the SentencePiece tokenizer described in section 7.1 and cut the texts by border characters (i.e., “<” and “>”) between an NE class token, such as “<PERSON>” and other strings. Then, each \( e_{\text{NE}} \) was assigned to its corresponding subword and NE class tokens were removed.

After preprocessing, we fine-tuned a model using BART\(^{12}\)\(\text{NE} \) architecture.

7.4 Evaluation Metrics

The following metrics were employed for evaluation.

**Document Classification**: We evaluated the outputs of document classification models using the macro F-measure of all classes.

**Headline Generation**: The outputs of headline generation models were evaluated using the F-measure of ROUGE-1 (R-1), ROUGE-2 (R-2), and ROUGE-L (R-L) (Lin, 2004), widely used as automatic evaluation metrics for headline generation. R-1 and R-2 are calculated based on overlap of uni-grams and bi-grams between a generated summary and its reference summary, respectively. Similarly, R-L is calculated based on overlap of the longest common subsequences between them.

8 Experimental Results

Table 5 shows the experimental results of document classification. Three models trained using NEs achieved higher F-measure than those without NEs. All models using NEs improved Recall.

<table>
<thead>
<tr>
<th></th>
<th>DC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
</tr>
<tr>
<td>BERT</td>
<td>0.7402</td>
</tr>
<tr>
<td>( \text{BART}^{\text{ENE}} \text{FNER} )</td>
<td>0.7375</td>
</tr>
<tr>
<td>( \text{BART}^{\text{ENE}} \text{FNER} )</td>
<td>0.7263</td>
</tr>
<tr>
<td>( \text{BART}^{\text{ENE}} \text{NNER} )</td>
<td>0.7348</td>
</tr>
<tr>
<td>( \text{BART}^{\text{ENE}} \text{NNER} )</td>
<td>0.7355</td>
</tr>
</tbody>
</table>

Table 5: Results of the document classification task. The bold fonts indicate better classification than the baseline BERT. P, R, and F1 denote the precision, recall, and F1-score, respectively.

We see from the results that NEs contribute to improve the score of document classification task with Transformer-based models.

Table 6 shows the results of the headline generation task. We also see that NE information contributed to improved accuracy of the headline generation task. Three models trained using NEs achieved higher accuracy than BART without NEs. The \( \text{BART}^{\text{ENE}} \text{NNER} \) showed the same accuracy as BART.

<table>
<thead>
<tr>
<th></th>
<th>HG (ROUGE F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R-1</td>
</tr>
<tr>
<td>BART</td>
<td>0.299</td>
</tr>
<tr>
<td>( \text{BART}^{\text{ENE}} \text{FNER} )</td>
<td>0.301</td>
</tr>
<tr>
<td>( \text{BART}^{\text{ENE}} \text{FNER} )</td>
<td>0.301</td>
</tr>
<tr>
<td>( \text{BART}^{\text{ENE}} \text{NNER} )</td>
<td>0.297</td>
</tr>
<tr>
<td>( \text{BART}^{\text{ENE}} \text{NNER} )</td>
<td>0.305</td>
</tr>
</tbody>
</table>

Table 6: Evaluation results on headline generation. The bold fonts indicate better accuracy than the baseline BERT. The underlined one is the same ROUGE score. R-1, R-2 and R-L indicate ROUGE-1, ROUGE-2 and ROUGE-L, respectively.

BERT\(^{\text{ENE}} \text{NNER} \) showed the best accuracy on document classification. BART\(^{\text{ENE}} \text{NNER} \) also showed the best accuracy on headline generation as in document classification. Our preliminary evaluation shows that NNER exhibited better accuracy than FNER. These results imply that accurate ENE information improved accuracy.

9 Related Work

There are several variants of BERT (Lan et al., 2020; Liu et al., 2019; He et al., 2020). Additionally, recent studies for neural document summarization focus more on pretraining methods (Lewis et al., 2020; Qi et al., 2020; Zhang et al., 2020). Al-
though we used BERT and BART, other methods can combine with the proposed method because the proposed method uses embeddings of NEs from finetuning, which is left as future work.

Marek et al. (2021) proposed an extractive summarization method that uses density of named entities to calculate the importance of a sentence. Furthermore, they proposed an abstractive summarization method that concatenates one-hot representation of named entity categories with token embeddings. Different from Marek et al. (2021), we investigate the effectiveness of the use of NEs in subword-based neural network models and different types of NEs, i.e., BNE and ENE. Additionally, we used embeddings to represent NEs instead of one-hot vectors to obtain further representation.

In addition to document summarization, some works use NEs for improving neural machine translation, which uses similar architectures as neural document summarization. Ugawa et al. (2018) proposed incorporating an additional LSTM layer to encode the sequence of NEs. Li et al. (2018) proposed inserting NE tags into the sequence of words in the source language. Different from Ugawa et al. (2018), we simply used embeddings for encoding NEs because Transformer-based models have self-attention mechanism that uses contextual information to obtain encoded results. Different from Li et al. (2018), our proposed method does not insert NE tags into the sequence of words because the time and memory complexities of Transformer-based models quadratically increase depending on the length of the sequence, which can be a significant problem in headline generation where a source document is long. Additionally, because subword tokenization is not used, their motivation differs from ours that investigates the effectiveness of NEs on subword-based neural networks.

Du et al. (2015) investigated the use of NEs in non-neural network models on document classification. While they reported use of NEs improve the accuracy of document classification, the contribution to subword-based neural network models was not investigated. Pivovarova and Yangarber (2018) compared the representation of NEs for neural network-based models in document classification task. They reported that replacing tokens of named entities with special tokens representing NE categories does not improve the accuracy of document classification. Our experiments showed that combining NE and token embeddings improved the accuracy of document classification.

10 Conclusion

This paper explored the effectiveness of NE information in large-scale pretraining models. We evaluated the effectiveness of NEs in document classification and headline generation tasks. The experimental results showed that NE information improved the accuracy of large-scale SOTA pretraining-based models. We incorporated NE information to pretrained models trained from subword sequences. For future work, we look to explore pretraining methods from subword sequences annotated with NE information from scratch.
References


Appendix

A Preprocessing for Data Sets

A.1 Mainichi News Articles
We pre-processed the main texts of Mainichi Shim-bun in five steps.

1) We remove articles with no main contents due to copyright.

2) We normalized the texts based on normalization form compatibility composition (NFKC) with `unicodedata.normalize` function of Python 3.6.9.

3) “<” and “>” in normalized texts were replaced by “{“ and “}”, respectively. This character-replacing process was needed to distinguish the original and attached characters by attaching the NEC label to text.

4) The texts were attached NEs in XML format with two NE recognizers.

5) We extracted NEs from annotated texts in BRAT\(^\text{13}\) format, then we re-attached NEs for the texts because the NE recognizer rarely changed the text.

B Excepted Publication Side Classes

Four classes of Mai-news were excluded (i.e., the four classes are front page (01), second page (02), third page (03), and the unknown genre that is not explained (27)) from the 18 classes on only document classification task because the classes were not related to the content of the article.

C Hyper Parameters

Table 7 shows hyperparameters of our experiments on document classification and headline generation.

\(^\text{13}\)https://brat.nlplab.org/
<table>
<thead>
<tr>
<th>Hyper Parameter</th>
<th>Document Classification</th>
<th>Headline Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimizer</td>
<td>AdamW (Loshchilov and Hutter, 2019)</td>
<td></td>
</tr>
<tr>
<td>Model Size</td>
<td>BASE</td>
<td>LARGE</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>2e-5</td>
<td>3e-5</td>
</tr>
<tr>
<td>Batch Size</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>Update Freq</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>Max Input Token Length</td>
<td>512</td>
<td>1,024</td>
</tr>
<tr>
<td>Max Epoch</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Number of Target Classes</td>
<td>14</td>
<td>-</td>
</tr>
<tr>
<td>Warm Up Step</td>
<td>0</td>
<td>500</td>
</tr>
<tr>
<td>Loss Function</td>
<td>cross entropy</td>
<td>cross entropy</td>
</tr>
<tr>
<td>Learning Rate Scheduler</td>
<td>linear decay</td>
<td>polynomial decay</td>
</tr>
<tr>
<td>Validation Freq.</td>
<td>1,000</td>
<td>20,000</td>
</tr>
<tr>
<td>#Max Train Data</td>
<td>-</td>
<td>50,000</td>
</tr>
<tr>
<td>#Max Test Data</td>
<td>-</td>
<td>10,000</td>
</tr>
<tr>
<td>#Max Dev Data</td>
<td>5,000</td>
<td>3,000</td>
</tr>
<tr>
<td>Beam Search Size</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>Upper Limit of Length of Token Generation</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 7: The hyperparameters in our experiments “#Max Train/Test/Dev Data” show the maximum numbers of data that were used in each experiment (“-” means no upper limit). “Validation Freq.” show how many time to process batch size data per one validation in the training phase. The model size of BASE and LARGE indicate that we used BERT BASE and BART LARGE, respectively.
TREMoLo-Tweets: a Multi-Label Corpus of French Tweets for Language Register Characterization

Jade Mekki\textsuperscript{1,2} Gwénaël Lecorvé\textsuperscript{1,3} Delphine Battistelli\textsuperscript{2} Nicolas Béchet\textsuperscript{4}

\textsuperscript{1}Univ Rennes, CNRS, IRISA / Lannion-Vannes, France
\textsuperscript{2}Universtité Paris Nanterre, CNRS, MoDiCo / Nanterre, France
\textsuperscript{3}Orange Labs / Lannion, France
\textsuperscript{4}Université de Bretagne Sud, CNRS, IRISA / Vannes, France

firstName.lastName @ \{iris.a.fr, orange.com, parisnanterre.fr\}

Abstract

The casual, neutral, and formal language registers are highly perceptible in discourse productions. However, they are still poorly studied in Natural Language Processing (NLP), especially outside English, and for new textual types like tweets. To stimulate research, this paper introduces a large corpus of 228,505 French tweets (6M words) annotated in language registers. Labels are provided by a multi-label CamemBERT classifier trained and checked on a manually annotated subset of the corpus, while the tweets are selected to avoid undesired biases. Based on the corpus, an initial analysis of linguistic traits from either human annotators or automatic extractions is provided to describe the corpus and pave the way for various NLP tasks. The corpus, annotation guide and classifier are available on \url{http://tremolo.irisa.fr}.

1 Introduction

Language registers are of particular interest in (socio-)linguistics because (1) they are a highly perceptible characteristics of discourse productions; (2) they represent a significant source of information about the writer/speaker, the relationship between interlocutors, or other elements of the communication context; (3) they are a concept known to all (advantage when running perceptual tests). Among the possible perceptions of this phenomenon, the partitioning into casual, neutral, and formal registers is probably the most used as it is found in many situations of everyday life. While corpora like GYAFC—where these variations are referred to as “formality level”—have recently popularized the domain (Rao and Tetreault, 2018), it is still poorly studied overall in Natural Language Processing (NLP), especially outside English. Moreover, current work largely focuses on textual types for which registers are already known from the linguistic literature\textsuperscript{1} whereas many new types, with their peculiarities, arise from the Computer-Mediated Communications (CMCs) (e.g., SMS, tweets...). Therefore, the analysis of CMC corpora in terms of language registers is a challenge both in terms of descriptive linguistics and applications in NLP.

As part of the TREMoLo project focusing on language registers\textsuperscript{2}, this paper tries to go beyond these limits and presents the corpus TREMoLo-Tweets, gathering 228,505 tweets (6M words), in French, with multi-label annotations among the casual, neutral and formal registers. The annotations come from a CamemBERT (Martin et al., 2020) model fine-tuned on a manually annotated subset (a.k.a. seed) of the whole corpus.

After a state of the art in Section 2, the corpus creation is presented in Section 3. Then, Section 4 provides first linguistic conclusions derived from statistics on manually and automatically-derived linguistic traits. Finally, possible tasks opened by the proposed corpus are listed in the conclusion.

2 Background and Motivation

Notion of registers. In sociolinguistics, the notion of language registers refers to the linguistic varieties associated with particular communication situations (Todorov, 2013). A key idea is that a language register can be characterized by specific patterns (Ferguson, 1982; Ledegen and Léglise, 2013). While the use of “level”, “style” or “genre” co-exist (Gadet, 1996; Bourquin, 1965; Joos, 1967), the term “register” tends to prevail (Biber, 1991; Sanders, 1993; Ure, 1982). Based on these points, we use the term “register” defined as a variation of linguistic forms, at different levels of analysis.

\textsuperscript{1}For instance, classically, insults are associated with the casual register while long sentences with subordinates are associated with the formal one.
\textsuperscript{2}\url{https://tremolo.irisa.fr}

Proceedings of Recent Advances in Natural Language Processing, pages 954–962
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_109
of the language, with respect to a given standard. This standard corresponds to the intersection of an “objective norm” (the grammatical rules) and a “subjective norm” (the rules of actual usage) (Gadet, 2007). Following this definition, a text is considered as formal when it completely conforms to the objective and subjective norms, neutral when it partially conforms to both, and casual when the objective norm is not followed.

Related work. In (Biber and Conrad, 2019; Biber, 1991), the use in the corpus of a priori defined linguistic features is quantitatively studied according to different axes: oral vs. written, formal vs. informal, etc. The purpose is to identify feature co-occurrences according to these axes.

For English, (Peterson et al., 2011; Pavlick and Tetreault, 2016) propose techniques to classify texts into formal vs. informal from a corpus of emails while (Sheikha and Inkpen, 2010) uses regression to predict a level of formality from a corpus of formal/informal texts.

For French, in (Lecorvé et al., 2019), the authors jointly study a classification task and the construction of a corpus of web pages3 annotated using an iterative semi-supervised approach.

The quality of the previously mentioned annotated corpora can be questioned from the perspective of language registers because (1) the composition of these corpora shows different biases by mixing text types or restricting the topics to a particular domain, (2) the manual annotations do not follow an annotation guide. In this paper, we propose to address these issues by (1) only focusing on tweets with a large range of domains, and (2) providing an annotation guide that is grounded on a linguistic analysis of language registers and CMCs.

Why choosing tweets? The constitution of a corpus of written texts representative of the real use of language registers presents two major difficulties. First, the strong link between some registers and some types of texts (e.g., the formal register associated with novels of classical literature, the casual register with discussion forums, and the neutral one with journalistic dispatches). Second, the oral and written modalities are intuitively associated with the casual and formals registers, respectively (Gadet, 2000; Rebourcet, 2008). To address these issues, CMCs—which are defined as “any human communication that occurs through the use of two or more electronic devices” (McQuail, 2010)—were chosen as their instantaneous nature can cause a “spoken-writing” style (or so-called “parlécrit” in French; Jacques (1999)). More precisely, Tweets are selected since they are CMCs and have a 280-characters limit, imposed by Twitter, which homogenizes the framework. The rather short length of tweets also prevents from texts where several registers could be present but not mixed (i.e., two distinct portions of a long text).

3 Corpus Creation

The corpus TREMoLo-Tweets is drawn from tweets collected in such a way as to cover the targeted spectrum of language registers while minimizing some unwanted biases. After various filterings and cleanings, a subset of these tweets was manually labeled. From a portion of these labeled tweets (training set), a CamemBERT classifier was fine-tuned to generalize the labels to the whole corpus. The result was validated on another part of the manually annotated tweets (test set). This section details the collection of the tweets, the labeling process, and the experimental validation.

3.1 Collection of the Tweets

Tweets have been collected by submitting queries to the Twitter API. Hence, the design of these queries is a key aspect. Here, the chosen strategy relies on the trending topics—which are the most used hashtags at a given time. Since they refer to striking events that are commented on by many users, we believe that they cover many different language functions and registers. Moreover, the varied nature of these events leads to equally diverse domains, which should enable to separate the notions of register and topic. In complement, the tweets were restricted to an unique geographical area (Paris) to minimize the impact of potential dialects. Tweets were collected on 10 different dates over a period of one month (August, 2020). For each date, 2,000 tweets were retrieved on average for each of the 50 top trending topics on that day.

Non-French tweets were removed using the langdetect Python library. Tweets with a probability < 0.9 for French were discarded. This arbitrary value is fixed in order to keep texts with the presence of some interesting non-French terms (e.g., “lol”, “stan”). Truncated tweets were removed by spotting the “horizontal ellipsis” characters. The
corpus counts 228,505 tweets (6,201,339 words). It has been standardized by the CamemBERT tokenizer, and morphosyntactically annotated by Talismane (Urieli and Tanguy, 2013).

3.2 Labeling of a Seed

Out of the entire corpus, 4,000 tweets have been randomly selected to be manually annotated in language register (named the seed). In the remainder, these tweets are referred to as the seed. Possible labels are the targeted registers (casual, neutral, formal) and an extra one to identify tweets that are badly encoded or incomprehensible. Multiples labels can be given to one tweet to reflect the co-presence of several registers. The objective is that each tweet of the seed is annotated with a degree of belonging for each of the 4 considered classes (i.e., summing to 100%).

Annotation guide. An annotation guide is built to frame the annotators’ work and, hence, the consistency of the final corpus. To do so, it defines the considered registers, following the principles outlined in Section 2, and gathers a set of linguistic descriptors interesting for the analysis of this corpus. These descriptors (detailed in Section 4.2) reflect peculiarities from the literature about language registers as well as CMCs. It is important to highlight that the annotation guide does not link the descriptors with specific registers. This is just a way to suggest potentially interesting aspects to be looked at. The annotator must then justify her/his labeling by selecting at least one of these linguistic descriptors. This annotation guide is given in the supplementary material (in French).

Manual annotation. The labeling of the seed has involved 4 experts such that each tweet has been annotated by 2 of them. For a given tweet, each annotator must indicate which register(s) (at least one) is (are) present and rank them according to their predominance. These choices had to be justified by at least one descriptor from the annotation guide. These annotations are released with the corpus, and their analysis is provided in Section 4.

In a post-processing phase, rankings are converted into degrees of belonging. For a given tweet, let $R$ denote the set of registers $r$ reported as present, $rank(r)$ the rank of each of them,

and its backward counterpart as $rank^{-1}(r) = 1 + Card(R) - rank(r)$. Then, the degree of $r$ is defined as the backward rank normalized by the sum of all ranks, i.e.:

$$degree(r) = \frac{\sum_{r \in R} rank^{-1}(r)}{\sum_{1 \leq i \leq Card(R)}^1}$$

(1)

To illustrate this conversion, let one consider that a tweet labeled with the neutral register as rank 1, and casual as rank 2. Then, the resulting degrees would be $\frac{2}{3} = 67\%$ and $\frac{1}{3} = 33\%$ for the neutral and casual registers, respectively. The degree would be 0 for the two others (formal and bin).

Agreement/disagreement between annotators. Given that all tweets are annotated by 2 experts, only those which are proposed by both of them are considered, and their degree is the average of the degrees from each annotator. In 976 tweets, the 2 annotators totally disagree (i.e. no shared label). Then, a third annotation is done by a new external annotator, and a given label is kept as soon as 2 annotators out of the 3 propose it. If no agreement can still be found for some tweets, they are discarded. Finally, 3,269 tweets remained.

Overall, the agreement between annotators is more significant for the casual and neutral registers (73% and 76%, respectively) than for the formal register (36%). Regarding the bin register, the agreement is perfect (100%). In detail, it appears that (1) for the casual register most of the divergences are with the neutral register, (2) for the neutral register with the casual register, and (3) for the formal register with the neutral register.

Overview of the annotated seed. The results of the manual annotation are dominated by the neutral register (51% of the seed, i.e., 1,698 tweets), followed by the casual (39%, i.e., 1,345 tweets), the formal (10%, i.e., 340 tweets), and finally the bin (almost 0%, i.e., 18 tweets). On average, when a label is present, its degree is high: 87% for the neutral register, 92% for the casual register, 87% for the formal register, and 98% for the bin register. Only 131 tweets have at least 2 registers present, against 3,138 with a single register. Even if the agreement policy is playing a role, this result shows that the tweets are not very nuanced in terms of registers. The short length can explain this phenomenon.

---

4The accuracy is 88.5% on the French TreeBank.
5Ph.D. students or researchers from CMCs or NLP.
6Equal ranks are permitted.
3.3 Automatic labeling of the whole corpus.

To label the full corpus in registers, a CamemBERT model\(^7\) is chosen to perform multi-label classification. In a first time, this model is fine-tuned on the sole seed. fine-tuned on 90% of the manually labeled seed. The idea is to use this model to label the whole corpus, and select some of these automatically annotated tweets in order to augment the model’s training data. Then, a new fine-tuning is performed and the whole corpus is definitely labeled. This section focuses on data preparation, the filtering strategy, and the experimental results.

Multi-label classification from the seed.

Multi-label classification is preferred to multi-dimensional regression to help the model distinguish strong differences between the registers. To do so, the degrees of belonging to each register are converted into binary labels. Tweets are labeled with a given register if and only if the associated degree is greater than or equal to 50%. The model is fine-tuned of 90% of the seed while the remaining 10% are for the evaluation. The model is the CamemBERT base version. Fine-tuning is performed with learning rate of \(10^{-4}\) and during 8 epochs. As a result, F1 values obtained for the casual, neutral, formal, and bin classes are 0.85, 0.84, 0.95, and 0.99, respectively.

Training data augmentation. To improve the performance, the training set from the seed is augmented by selecting automatically labeled tweets from the non-seed part of the corpus (i.e., the other remaining tweets). This is implemented by filtering the tweets for which one of the predicted labels is a probability greater than a threshold \(T_1\). The label probabilities of the selected tweets are then binarized in the same way as the seed, and a new fine-tuning is performed based on the augmented training data. Figure 1 shows the F1 values after this second training. These results demonstrate that data augmentation is worth it, and rather robust across the values for \(T_1\) (all values range in \([0.95, 1]\)). Best values for \(T_1\) seem to range in \([0.9, 0.99]\). Percentages below each point refer to the proportion of the tweets labeled for each register in the whole corpus. It appears that data augmentation did not really change these proportions compared to the distribution in the seed.

To deepen this study of robustness, another series of experiments was conducted to test the number of new samples required to efficiently perform data augmentation. This is noted as another threshold \(T_2\) on the number of tweets added. To do so, label probabilities provided by the initially trained classifier are sorted in descending order. The conclusions from these experiments are that 6% (about 14,000 tweets) of the whole corpus is enough to obtain good labeling results.

These various results and their stability, which follow the trend in the manual labels, tend to indicate that the final labels of the whole corpus are of good quality.

4 Linguistic analysis

This section presents a first linguistic analysis of the corpus TREMoLo-Tweets in terms of language registers. After presenting the underlying linguistic descriptors, this section reports which of these descriptors have mostly been selected by the annotators, and how they appear on the whole corpus using systematic automatic extractions.

4.1 Linguistic descriptors

A set of 47 linguistic descriptors is made by updating a list, from a study that has already identified features in the scientific literature on language registers (Mekki et al., 2018), with those specific features to CMCs (examples Table 1). Among the 47 descriptors: 15 are syntactic, 9 morphological, 9 lexical, 6 discursive, 5 lexico-syntactical, and 3 phonological. They were chosen to help the annotators make their decision, and assert that their labeling is motivated. Among them, some elements are specific to tweets: \(\text{(Paveau, 2013)}\) calls them “technomorphems”\(^8\). One of our contributions is to integrate them instead of discarding them as in (Go et al., 2009; Pak and Paroubek, 2010; Agarwal et al., 2011). The main technomorphems are:

---

\(^7\)Simpler models were tried but obtained lower accuracy.

\(^8\)Forms that arise from digital discourses.
Table 1: Ratio of usage of each linguistic descriptor in the justifications of annotators when manually labeling the seed. Descriptors with all ratios lower than or equal to 5% are not reported.

<table>
<thead>
<tr>
<th>ID</th>
<th>Linguistic descriptors</th>
<th>Casual</th>
<th>Neutral</th>
<th>Formal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Absence of classic final punctuation</td>
<td>63%</td>
<td>40%</td>
<td>3%</td>
</tr>
<tr>
<td>2</td>
<td>Idiomatic expression</td>
<td>41%</td>
<td>27%</td>
<td>1%</td>
</tr>
<tr>
<td>3</td>
<td>Absence of an expected item</td>
<td>35%</td>
<td>16%</td>
<td>1%</td>
</tr>
<tr>
<td>4</td>
<td>Modalizing expression</td>
<td>33%</td>
<td>23%</td>
<td>11%</td>
</tr>
<tr>
<td>5</td>
<td>Electronic spelling</td>
<td>27%</td>
<td>2%</td>
<td>0%</td>
</tr>
<tr>
<td>6</td>
<td>Contiguous repetition of items</td>
<td>25%</td>
<td>5%</td>
<td>1%</td>
</tr>
<tr>
<td>7</td>
<td>Shortening of words</td>
<td>23%</td>
<td>9%</td>
<td>2%</td>
</tr>
<tr>
<td>8</td>
<td>Foreign language borrowing</td>
<td>19%</td>
<td>8%</td>
<td>3%</td>
</tr>
<tr>
<td>9</td>
<td>Removal of certain letters due to elision or apocope</td>
<td>17%</td>
<td>2%</td>
<td>0%</td>
</tr>
<tr>
<td>10</td>
<td>“ça” preferred to “cela”</td>
<td>16%</td>
<td>10%</td>
<td>2%</td>
</tr>
<tr>
<td>11</td>
<td>Interjection</td>
<td>14%</td>
<td>1%</td>
<td>1%</td>
</tr>
<tr>
<td>12</td>
<td>Oomatopoeia</td>
<td>11%</td>
<td>1%</td>
<td>0%</td>
</tr>
<tr>
<td>13</td>
<td>Character repetition</td>
<td>11%</td>
<td>1%</td>
<td>0%</td>
</tr>
<tr>
<td>14</td>
<td>“s’est-ce que” for interrogative sentences</td>
<td>9%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>15</td>
<td>Capital letters used outside their conventional usage</td>
<td>9%</td>
<td>3%</td>
<td>1%</td>
</tr>
<tr>
<td>16</td>
<td>“alle” for the construction of the future tense</td>
<td>8%</td>
<td>7%</td>
<td>1%</td>
</tr>
<tr>
<td>17</td>
<td>Discursive termination</td>
<td>7%</td>
<td>1%</td>
<td>1%</td>
</tr>
<tr>
<td>18</td>
<td>Verlan (i.e. reversing the terms syllable by syllable)</td>
<td>6%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>19</td>
<td>“est-ce que” for interrogatives sentences</td>
<td>6%</td>
<td>19%</td>
<td>0%</td>
</tr>
<tr>
<td>20</td>
<td>Present as the only tense used</td>
<td>12%</td>
<td>14%</td>
<td>4%</td>
</tr>
<tr>
<td>21</td>
<td>Hashtag syntactically independent</td>
<td>9%</td>
<td>9%</td>
<td>3%</td>
</tr>
<tr>
<td>22</td>
<td>Hashtag with no syntactic relation</td>
<td>3%</td>
<td>7%</td>
<td>8%</td>
</tr>
<tr>
<td>23</td>
<td>Doubled element</td>
<td>3%</td>
<td>6%</td>
<td>3%</td>
</tr>
<tr>
<td>24</td>
<td>Diversity of verbal tenses</td>
<td>2%</td>
<td>25%</td>
<td>62%</td>
</tr>
<tr>
<td>25</td>
<td>Several sentences with classical punctuation</td>
<td>2%</td>
<td>25%</td>
<td>56%</td>
</tr>
<tr>
<td>26</td>
<td>Hashtag syntactically integrated</td>
<td>3%</td>
<td>19%</td>
<td>49%</td>
</tr>
<tr>
<td>27</td>
<td>Presence of the double negation</td>
<td>3%</td>
<td>18%</td>
<td>41%</td>
</tr>
<tr>
<td>28</td>
<td>“vous” preferred to “nous”</td>
<td>1%</td>
<td>9%</td>
<td>19%</td>
</tr>
<tr>
<td>29</td>
<td>Presence of subject/verb inversion</td>
<td>3%</td>
<td>8%</td>
<td>38%</td>
</tr>
<tr>
<td>30</td>
<td>Speech citation</td>
<td>1%</td>
<td>18%</td>
<td>28%</td>
</tr>
<tr>
<td>31</td>
<td>Presence of the user’s identifier integrated in a phrase</td>
<td>1%</td>
<td>12%</td>
<td>26%</td>
</tr>
<tr>
<td>32</td>
<td>Presence of subject/verb inversion</td>
<td>0%</td>
<td>3%</td>
<td>20%</td>
</tr>
<tr>
<td>33</td>
<td>Diversity of logical connectors</td>
<td>0%</td>
<td>4%</td>
<td>20%</td>
</tr>
<tr>
<td>34</td>
<td>Pictogram that highlights information</td>
<td>1%</td>
<td>3%</td>
<td>17%</td>
</tr>
<tr>
<td>35</td>
<td>Pictogram in the replacement function</td>
<td>1%</td>
<td>3%</td>
<td>6%</td>
</tr>
</tbody>
</table>

The hashtags, and the pictograms.

**Hashtags** are defined as one or more contiguous words preceded by a # sign (e.g. “#MerryChristmas”). Some typologies of hashtags emphasize their indexing function (Jackiewicz and Vidak, 2014) (e.g., “#Tokyo2020”). In addition to this, we assume that their syntactic integration, that is their use as a standard lexeme, also brings variety to the language registers.

**Pictograms** refer to both “emoticon” and “emoji”. The (Mauguet al., 2020)’s typology of 3 functions has been used and adapted to our analysis: (1) the replacement function (when a pictogram replaces a syntagm); (2) the illustration function (when it has a referential function); (3) the modalization function, when it indicates the emotion or the cognitive position of the author wrt to his/her statement. Then a 4th function has been added: (4) the framing/structuring function (when a pictogram surrounds or points at information).

### 4.2 Human justifications on the seed

For a given tweet, a descriptor is manually selected by an annotator when it mainly motivates the attribution of a register.

The casual register seems to be marked by the absence of classical punctuation (#1), idiomatic expressions (#2), and modalization expressions (#4). Here, the expressive role of the absent punctuation seems to have been taken over by other linguistic objects (e.g., pictograms).

The neutral register is marked by the absence of all the linguistic descriptors.

Then a 4th function has been added: (4) the framing/structuring function (when a pictogram surrounds or points at information).
of classical punctuation (#1), the diversity of the verbal tenses (#28), and the presence of several sentences with classical punctuation (#29). The neutral register seems less clear-cut (notably with different uses of the punctuation marks).

The formal is also characterized by the presence of several sentences with classical punctuation marks (#29), the syntactic integration of hashtags (#30), and pictograms that highlight information (#38). Therefore, technomorphemes in the formal registers show that CMC-specific items have been integrated into the French standard.

### 4.3 Automatic extraction on the whole corpus

In order to analyze the whole corpus, symbolic rules were implemented to automatically spot the presence of the linguistic descriptors. Let one note that 5 descriptors could not be implemented since they refer to complex notions. This automatic extraction is not selective (all descriptors present are taken from the tweet) unlike the manual extraction from the seed which is selective (only descriptors that contribute the most to the register are taken from the tweet). The overview of these exhaustive extractions is provided in Table 2.

To characterize a register $r$ according to the other registers (noted $o$), the importance of each descriptor noted $d$ observed in $r$ is measured by computing a growth rate ($GR$) as the ratio between relative frequencies of $d$ in $r$ as opposed to $o$:

$$GR(d, r, o) = \begin{cases} \infty, & \text{if } f_d(r) = 0 \\ \frac{f_d(r)}{f_d(o)}, & \text{otherwise} \end{cases},$$

(2)

where $f_d$ denotes the relative frequency in a register as reported in Table 3. The relative frequencies for the register “other” ($f_d(o)$) is computed by merging all tweets that are not of register $r$. If $GR(d, r, o) > 1$, $d$ is considered as emergent.

Table 3 reports for each register the descriptors with the highest growth rates. Interestingly, some rare descriptors appear whereas they were previously skipped in Table 2 ($\leq 5\%$). Then, it appears that the growth rates are lower for the neutral register than for the casual and the formal ones. This shows the fuzzy limits with the other registers. On the contrary, the casual register has high values, which means that it is characterized by unambiguous specific traits. The presence of technomorphemes in the emergent descriptors, for the casual and formal registers, confirms the integration of the Twitter-specific elements to the French standard. However, they are used differently by register.

For the neutral register, a commercial application uses hashtag indexing functions:

Le jeu #MonstrumGame de @X sort [...] le 23 octobre (@X’s #MonstrumGame comes out [...] on October 23rd).

The pictogram seems to replace classic punctuation marks at the end of the sentence:
<table>
<thead>
<tr>
<th>ID</th>
<th>Casual GR C vs. Others Example (translation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Electronic spelling 7.00 / 1.00% Ha allez ooooh !!! (Eh la lillll)</td>
</tr>
<tr>
<td>18</td>
<td>“if” replaced by “y” 6.00% / 2.00% Y’en a le 25 (Thr’s some the 25)</td>
</tr>
<tr>
<td>40</td>
<td>Pattern “juste” 0.50% / 0.20% Juste comme ça (Just like that)</td>
</tr>
</tbody>
</table>

Neutral GR N vs. Others Example (translation)

<table>
<thead>
<tr>
<th>ID</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Absence of an expected item 1.50 / 0.07% ils vont quand même pas (Still not)</td>
</tr>
<tr>
<td>16</td>
<td>Capital letters used outside their conventional usage 31.00% / 29.00% 17 juillet pour OM DÉVELOPPEMENT (July 17 for OM DEVELOPMENT)</td>
</tr>
<tr>
<td>25</td>
<td>Hashtags syntactically independent 12.00% / 11.50%</td>
</tr>
</tbody>
</table>
| 34 | Text structured by punctuation 15.00% / 7.00% | VIDEO. Crise des transports : (VIDEO. Traffic crisis :)

Formal GR F vs. Others Example (translation)

<table>
<thead>
<tr>
<th>ID</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>Pictogram that highlights information 2.20 / 0.30% #X banni de #Facebook</td>
</tr>
</tbody>
</table>
| 34 | Text structured by punctuation 15.00% / 7.00% VIDEO. Crise des transports : (VIDEO. Traffic crisis :)
| 30 | Hashtag syntactically integrated 11.00% / 5.00% #ViolencesPolicieres ne sont pas (Violence experienced in #France aren’t #incivilites) |

Table 3: Top 3 automatic descriptors w/ highest growth rate for each register against the others in the whole corpus.

@X @X je pense que ça fait référence à des dates de sortie 😃. En septembre ça tombe sur des Vendredi (I think it refers to release dates 😃. In September it falls on Fridays)

For the formal register, hashtags are syntactically integrated:

Les violences vécues en #France ne sont pas des #incivilites (Violence experienced in #France aren’t #incivilites)

The pictograms are used with their framing/structuring function which brings a kind of verticality to the tweet:

🔥 L’ #Amazonie brûle ! 🔥 Partout en France, les citoyen.nes aux côtés de @X demandent des actes au gouvernement Macron […] (🔥 The Amazon is burning ! 🔥 Everywhere in France, citizens alongside @X demand actions from the Macron government […]

The casual register seems more used for dialogue between users and pictograms are used for their modalization function to provide extra-linguistic information: they compensate for the lack of paraverbal information such as prosody.

"@X La France part en couille et l autre con jardine au Liban 😂😂" (@X France is going to the dogs and the other idiot is gardening in Lebanon 😂😂)

Moreover, marks of orality are found:

@X Bah là j’ai pas encore test le son mais en tout cas niveau confort y’a pas photo […] (Bah there I did not test the sound yet but in any case level of comfort it’s not photo […] )

Thus, these first analyses highlight the corpus quality, and the relevance of the set of linguistic descriptors for CMC data. Likewise, the analysis of the registers identifies different linguistic functions on Twitter (argumentative, commercial, conversational speech).

5 Conclusion

In this paper, we presented the corpus TREMoLo-Tweets which gathers 228,505 tweets labeled in casual, neutral and formal registers. For this purpose, a seed was manually annotated with multiple labels, following an annotation guide derived from a linguistic analysis of the corpus. Using a CamemBERT model and data augmentation, the whole corpus is entirely labeled with an experimentally demonstrated high reliability. Furthermore, statistics on linguistic descriptors are reported to demonstrate the richness of the corpus.

The labels, linguistic descriptors, and the large size of the corpus pave the way to future tasks:

- **Standard NLP tasks on a seldomly studied style factor.** Classification (predicting the registers of a given text) and natural language generation (style transfer).

- **Data mining.** How to rank the descriptors to discriminate the registers against each other? How to reconstruct the features of interest from the raw words? Justifications given by the annotators can be used as a reference.

- **Interdisciplinary work.** Discovery of new fundamental knowledge about language registers by crossing NLP and sociolinguistics, like in (Abitbol et al., 2018) where study the linguistic variations are studied according to the writers’ geographical areas and economic social status.

Acknowledgements

This work was supported by the TREMoLo project (ANR-16-CE23-0019) of the National Research Agency (ANR).
References


Denis McQuail. 2010. McQuail’s mass communication theory. Sage publications.


Abstract

Online reviews are an essential aspect of online shopping for both customers and retailers. However, many reviews found on the Internet lack in quality, informativeness or helpfulness. In many cases, they lead the customers towards positive or negative opinions without providing any concrete details (e.g., very poor product, I would not recommend it). In this work, we propose a novel unsupervised method for quantifying helpfulness leveraging the availability of a corpus of reviews. In particular, our method exploits three characteristics of the reviews, viz., relevance, emotional intensity and specificity, towards quantifying helpfulness. We perform three rankings (one for each feature above), which are then combined to obtain a final helpfulness ranking.

For the purpose of empirically evaluating our method, we use review of four product categories from Amazon review

1

1 http://jmcauley.ucsd.edu/data/amazon/

often no such ranking options. Some websites rank reviews based on the posting date or rating star, for example, Trustpilot.com and Reviews.io. Amazon uses a crowdsourcing mechanism, a voting system, to gather feedback on review helpfulness, and then rank them based on the overall votes they received (Amazon.com). A user can vote for a review as being helpful or unhelpful. Amazon was estimated to receive a revenue of about $2.7 billion by providing simple question “was this review helpful to you?” (Spool, 2009).

Although such a voting system is helpful for customers, it has several limitations due to the inherent character of the voting process. There are number of reasons: 1) not all reviews get the helpfulness vote; 2) the helpfulness voting does not work for cold star review (i.e., a new user or a new review will have much less votes) (Singh et al., 2017); 3) reviews receiving helpfulness votes would tend to gather more vote due to the snowball effect (e.g., phenomena such as social proof (Cialdini, 1987)).

In this work we hypothesise that helpfulness of a review should be assessed based on three characteristics, namely relevance (whether the review discusses the key features relevant to a specific product), emotional intensity (level of emotions expressed within a review) and specificity (level of details discussed in a review). We motivate the importance of each of those features later in the paper.

We then propose an unsupervised helpfulness ranking method that does not depend on the helpfulness votes and only takes under consideration the content of the review and the star rating. We demonstrate that our proposed method outperforms the state-of-the-art review ranking techniques, through an extensive empirical evaluation.

The paper is organised as follows. In the next section we present an overview of the work that has been carried out in this space. Following this, we provide the motivation and technical details of
the proposed method. Finally, the results of the experimental evaluation are demonstrated followed by the discussion.

2 Related Work

Several approaches to automatically determining the helpfulness of online reviews have been explored in the past. In majority of the existing work, supervised machine learning models have been employed considering the problem as a predictive task (i.e., predict whether/how useful a review is) (Martin and Pu, 2014; Krishnamoorthy, 2015; Liu et al., 2017; Malik and Hussain, 2017; Singh et al., 2017; Wu et al., 2017; Enamul Haque et al., 2018; Lee et al.; Alsmadi et al., 2020). With supervised approaches, various types of features such as linguistic features (Krishnamoorthy, 2015; Malik and Hussain, 2017; Wu et al., 2017) or textual features (i.e., polarity, subjectivity, entropy and readability) (Singh et al., 2017; Lee et al.; Siering et al., 2018) are first extracted from the reviews, with machine learning methods used over such data to train a predictive model. In a few papers, unsupervised learning based approaches have been used to rank reviews based on their helpfulness or relevance (Tsur and Rappoport, 2006; Wu et al., 2011; Woloszyn et al., 2017). It is very apparent that the majority of work has been focused on using supervised machine learning and unsupervised learning has not been well explored in this space. Supervised learning methods depend on large, annotated datasets to train the model. Unfortunately, most of the publicly available online reviews datasets do not have labels related to their helpfulness. This makes the unsupervised learning based approaches much more attractive and hence it is the focus of our work.

A review ranking method based on unsupervised learning was proposed by Tsur and Rappoport (2006). The authors first created a corpus of core dominant terms for the reviews representing the key aspects relevant to a specific product. Dominant terms were obtained by computing the frequency of all terms in a reviews collection and re-ranking them by their frequency in the reference to the British National Corpus, a baseline corpus. They named the corpus as virtual core (VC) review and represented it as feature vectors. Following this, they ranked the reviews according to their distance from the virtual core review vector. They assumed that the smaller the distance between a review and the virtual core review, the more relevant/helpful the review is.

Wu et al. (2011) proposed a ranking method to detect low quality reviews by using link analysis techniques. Three ranking algorithms have been implemented in their study which are (1) PageRank algorithm (Page et al., 1999), (2) HITS algorithm (Kleinberg et al., 2011), and (3) Length algorithm. First, they construct a graph for each review of a product where the vertexes are sentences in a review. Two directional edges between two vertexes are induced if they are similar according to specific POS tag i.e., nouns, adjectives, and verb. They compute the centrality scores of sentences using the PageRank and the HITS algorithms. A score for each review was obtained by summing all the centrality scores of all the sentences in a review and then rank the review based on the high centrality scores. The Length algorithm was used to rank all reviews based on total number of words. They count the number of words for each review and rank it based on the high score. The authors conjecture that high-quality review should contain more words than poor reviews. Two baseline methods were used for comparison in their experimental evaluation. From the evaluation, it could however be observed that their results were only slightly inferior in comparison to the baselines.

Inspired by the work proposed in (Martin and Pu, 2014), Woloszyn et al. (2017) developed MRR (Most Relevant Review), a novel unsupervised algorithm to rank reviews based on their estimated relevance. MRR algorithm consists of three steps: (1) First, they construct a graph of reviews for each product where the nodes are the reviews and the edges are defined based on the similarity between pairs of reviews. Two similarity scores are considered: cosine similarity between TF-IDF vectors computed for each review, and similarity between rating scores of reviews (i.e., rating scores from 1 to 5 given by reviewers), (2) This is followed by graph pruning that works by removing all edges with the similarity scores lower than the minimum threshold value, (manually set as $\beta=0.8553$), (3) Finally, the centrality scores are calculated for each review using PageRank algorithm. The authors hypothesise that the more central reviews should be considered as most relevant. Two state-of-the-art unsupervised learning (Tsur and Rappoport, 2006; Wu et al., 2011) and two supervised learning methods (i.e., one of the method use the same features
as (Wu et al., 2011)) were adopted in the experimental evaluation for comparison. Although, their results were lower than those obtained by supervised learning methods, they outperformed the two unsupervised learning based approaches.

In this work, we propose a new unsupervised method for ranking online reviews based on their helpfulness. Apart from the relevance (as in case of the existing unsupervised techniques), our method also considers the emotional intensity and the specificity of the reviews while assessing their helpfulness; this makes it unlike any of the approaches discussed above. For the text representation, we apply the Roberta state-of-the-art language model as opposed to TF-IDF used by the existing unsupervised methods.

3 Methodology

The key novelty of the proposed method is that it incorporates three different characteristics of online reviews while ranking them according to their helpfulness. We hypothesise that the helpfulness of a review should be determined based on the following features:

a) Relevance. Relevance indicates how well a review matches with customer’s specific information needs (Liu et al., 2019). In other words, a helpful review should discuss the key features of a product, which are important for the future buyers (e.g., “The camera is easy to use, it is compact and perfect for travelling.”). Review’s relevance has been modelled by the existing work (Wu et al., 2011; Woloszyn et al., 2017) using graph composed of all reviews, their similarities and various centrality measures. It was assumed that the reviews that are the most central within the graph contain the most relevant information about the product. In our work, we take a similar approach, however, instead of graphs we used a simpler pair similarity based method.

b) Emotional Intensity. We hypothesise that emotions play an important part in a review process as they allow customers to express their feelings and experiences through opinions. Therefore, a good review should contain a good balance of both, facts and emotions. The relationship between helpfulness of online reviews and emotions have been explored by Malik and Hussain (2017) where they studied which emotions are important for helpfulness prediction. Martin and Pu (2014) used emotions to detect helpful reviews by applying different classification models (i.e., SVM, Random Forest, and Naive Bayes) and demonstrated that their approach outperformed methods using POS tagging features. Emotion information has not been considered by any of the existing unsupervised methods. In this work, we propose to consider the level of emotions contained within a review as one of the factors in determining their helpfulness.

c) Specificity. A review of a product will be considered as useful/informative if it discusses various features of the products. In other words, instead of just expressing satisfaction/dissatisfaction from a product (e.g. “I hate this camera and would not recommend it”), it is much more helpful if the review explains what good or bad there is about the product (e.g. “The battery life is too short and the zoom is rather poor.”). The greater number of different features is mentioned in a review, the more informative the review is for any potential buyer/customer. It should be noted that there is a distinct difference between the relevance and the specificity. With relevance, we assess whether the key characteristic of a product was discussed. While with specificity, we evaluate the level of details that was provided while discussing different features of a product. Following this reasoning, we propose to consider the number of different entities mentioned in the reviews while ranking the reviews based on their helpfulness. Such a specificity feature has also not been considered by any of the existing work.

Apart from the aforementioned characteristics, we also consider the star rating of the reviews in our ranking process. It has been demonstrated in the literature that the application of star rating is beneficial when evaluating the helpfulness of a review (Tsur and Rappoport, 2006; Schuff and Mudambi, 2010; Singh et al., 2017).

The pseudocode of our proposed methods is presented in Algorithm 1. The input to the method is a collection of reviews related to the same products. Each review contains the review text (r) and the star rating associated with this review (s). In the first step of the algorithm, the input reviews are ranked
separately on the basis of their relevance, emotional intensity and specificity. For the relevance ranking, we create a product-specific “summary document” (sum), which contains all individual reviews collated together. The summary document and each individual review are then converted into vectors using the RoBERTa pre-trained language model (Liu et al., 2019). For this part, any other embedding model (such as Word2Vec or Glove) can be considered. We used the RoBERTa model as it has recently received state-of-the-art results on many NLP benchmark datasets (Liu et al., 2019). Following this, the cosine similarity between each individual review and the summary document is calculated as its relevance score. It is worth noting that the proposed relevance ranking method is much simpler and faster than those of the baseline, which uses graphs to model similarity between reviews. With the second ranking, the reviews are ranked based on their emotional intensity. To identify different emotions in the reviews we used the DepecheMood++ (Araqe et al., 2018) lexicon that contains 187942 words with 8 emotions intensity value for each word; this could be replaced with any emotion lexicon. For each review, we first identify all words which are present in the lexicon. Following this, all the intensity values assigned to those words in the lexicon are added together. The final emotion score assigned to each review is the accumulation of intensity value by summing all emotion words within this review.

Finally, for the specificity ranking, we first apply name entity recognition and extract entities from the reviews using the NLTK library\(^2\). We calculate the specificity score for each review as the sum of all entities that it contains. All the reviews are then sorted separately based on the three scores. As the outputs of the aforementioned steps, we obtained three rankings of the reviews, which were constructed based on the relevance, emotional intensity, and specificity of the reviews (lines 15-17).

As mentioned earlier, we also consider the star rating in our ranking method as it is considered as an good indicator of reviews helpfulness (Tsur and Rappoport, 2006; Schuff and Mudambi, 2010; Singh et al., 2017). We process the star rating by calculating the absolute deviation. The use of star rating deviation as a feature has been demonstrated in (Jindal and Liu, 2008; Lim et al., 2010; Jiang et al., 2013; Xu, 2013; Savage et al., 2015; Saumya and Singh, 2018) and some of the authors apply absolute deviation for the star rating (Danescu-Niculescu-Mizil et al., 2009; Mukherjee et al., 2013a,b; Runa et al., 2017). First, we calculate the average of all star ratings of a product review (line 20). In the next step, for each review \(r_i\), we calculate its absolute deviation (RAD) from the average star rating as per Eq 1.

\[
AD_i = |s_i - \text{avg}|
\]

\[
RAD_i = (1 - \alpha) \ast AD_i
\]

where \(s_i\) is star rating for review \(r_i\), typically between 1 and 5. Finally we calculate the rating absolute deviation (RAD) (line 22) as per equation 1, where \(\alpha\) is used to balance the impact of the star rating on the final ranking and its value has been adopted from (Wooszyn et al., 2017), \(\alpha = 0.867168\).

The RAD value will be further included in the final ranking process together with the other three rankings as explained below. For combining the three rankings (i.e., relevance, emotional intensity and specificity), we applied the z-score minimization method (Standard score, 2021). First, the mean (\(\mu\)) and the standard deviation (\(\sigma\)) of the three ranking positions are computed for each review \(r_i \in R\). In the next step we calculate the z-score distance matrix calculating the z-score for each review and every possible ranking position according to the following formula (lines 21-26):

\[
z\text{-score} = |(p - \mu)/\sigma|
\]

where \(p\) is the proposed ranking position.

The intuition behind this is to find the most statistically best ranking position by minimizing the aggregate z-score distance globally. The idea is from (Du et al., 2019) where they used exhaustive process for all possible features combination to find the best combination for helpfulness prediction. However, instead of using exhaustive process, we use a faster approach. The rows of matrix represent the number of reviews for each product and the columns represent the number of possible positions in the ranking (i.e., this is a squared matrix). Each cell of the matrix \((c_{ij})\) contains a position score calculated for review \(r_i\) and position \(j\) using equation 2. The z-score tells us how far each of the proposed ranking positions is from the mean position of the review. We further add the previously calculated RAD value to the z-scores.
Algorithm 1: The proposed algorithm for ranking online reviews based on their helpfulness

Require: List of reviews and their star ratings $R = \{(r_i, s_i)\}_{i=1..n}$ related to a single product
Ensure: The reviews ranked according to their helpfulness

1: $\Join \_\text{review}$ = join all reviews in $R$
2: $\text{sum} = \text{convert} \ Join \_\text{review}$ into Roberta embedding
3: for each review $r_i$ in $R$ do
4: $r_i \_\text{embed} = \text{convert} \ r_i$ into Roberta embedding
5: $r_i \_\text{relevance} \_\text{score} = \text{CosineSimilarity} (\text{sum}, r_i \_\text{embed})$
6: for each word $w_j$ in $r_i$ do
7: if $w_j$ in DepecheMood++ then
8: $\text{emotion} \_\text{scores}[j] = \text{sum all emotions intensities of } w_j \text{ from DepecheMood++}$
9: end if
10: end for
11: $r_i \_\text{specificity} \_\text{score} = \text{count number of entities in } r_i$
12: end for
13: $\text{rank}_1 = \text{rank } R \text{ based on } \{r_i \_\text{relevance} \_\text{score}\}_{i=1..n}$
14: $\text{rank}_2 = \text{rank } R \text{ based on } \{r_i \_\text{emotion} \_\text{score}\}_{i=1..n}$
15: $\text{rank}_3 = \text{rank } R \text{ based on } \{r_i \_\text{specificity} \_\text{score}\}_{i=1..n}$
16: $\text{rank} \_\text{combine} = \text{combine all ranking } (\text{rank}_1, \text{rank}_2, \text{rank}_3)$
17: $\text{avg} \_\text{star} = \text{average of all star ratings } \{s_i\}_{i=1..n}$
18: for each $r_i$ in $R$ do
19: $RAD_i = (1 - \alpha) \ast |s_i - \text{avg} \_\text{star}|$
20: for $j$ in len($R$) do
21: $\text{position} \_\text{score}[i][j] = \alpha \ast |z \_\text{score}| + RAD_i$
22: end for
23: end for
24: for column in len(position) do
25: $\text{sum} \_\text{score} = 0$
26: for row in len(position) do
27: $\text{sum} \_\text{score} = \text{sum} \_\text{score} + \text{position}[\text{row}][\text{column}]$
28: for $j$ in len(position) do
29: $\text{sum} \_\text{score} = \text{sum} \_\text{score} + \text{position}[\text{row}][\text{column}]$
30: end for
31: $\text{total} \_\text{score} = \text{sum} \_\text{score} - \text{min} (\text{position}[\text{row}][\text{column}])$
32: end for
33: select column where $\text{total} \_\text{score} = \text{max} (\text{total} \_\text{score})$
34: assign review at the position where $\text{position} \_\text{score} = \text{min} (\text{position} \_\text{score})$
35: delete the column and row and repeat step 28-37 until convergence

In the matrix. The final step is to find out which set of ranking positions of the reviews gives the lowest total $z$-score distance. For this purpose we use an iterative solution (lines 28-37) which is explained below. For each column, we sum its values and subtract the minimum value from this column, obtaining a score referred to as $\text{total} \_\text{score}$. Then, we select the column with the maximum $\text{total} \_\text{score}$. After that, we find the minimum value in that column. The corresponding review is then assigned to the position. The next step is to delete the column and row and repeat the same for the rest of reviews until all the positions are filled. For instance, if the largest $\text{total} \_\text{score}$ is at column 4 and the minimum position score on that column belongs to review 1, then assign the review 1 to that position, i.e. 4 which is now the re-ranked position of review 1.

4 Experimental Evaluation

4.1 Datasets

For the purpose of this study, we use dataset from Amazon three reviews (from May 1996 – July 2014) for four categories of products, namely (1) Electronics, (2) Books, (3) CDs Vinlys and (4) Movies TV products, with raw data size of 1.48 GB, 9.46 GB, 1.33 GB, and 1.93 GB respectively. In this study, we only use four features: ASIN as a unique product id, ReviewText for performing the three rankings, Overall in order to include the rating star in the final ranking and Helpfulness Votes for the evaluation purposes. All the data has been processed and filtered according to the following steps. First, the product should have minimum 30 reviews. Each review should contain minimum four sentences. The review should have minimum five helpfulness votes. The details regarding the size of each dataset before and after pre-processing are listed in Table 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Books</td>
<td>8,898,041 rev</td>
<td>109,099 rev</td>
</tr>
<tr>
<td>Electronics</td>
<td>1,689,188 rev</td>
<td>8,134 rev</td>
</tr>
<tr>
<td>CDs and Vinyls</td>
<td>1,097,592 rev</td>
<td>11,448 rev</td>
</tr>
<tr>
<td>Movies &amp; TV</td>
<td>1,697,533 rev</td>
<td>31,035 rev</td>
</tr>
</tbody>
</table>

Table 1: Amazon dataset
4.2 Baseline and Evaluation

As a baseline, we implemented the state-of-the-art unsupervised ranking method MRR (Woloszyn et al., 2017), which has been described in Section 2. This is the most recent work that has been done in this space using unsupervised learning. In the original paper (Woloszyn et al., 2017), the results were also compared with two other unsupervised approaches and supervised models and it was demonstrated that MRR outperformed others baseline (Tsur and Rappoport, 2006; Wu et al., 2011). Therefore, we only use MRR as the baseline.

For further evaluation, we also explored different variants of our proposed methods. We considered using the summary of the reviews instead of their full content. The summaries of the reviews were first obtained with the SUMY library\(^4\) and then provided as an input to the algorithm described in Algorithm 1. We also evaluated the performance of our method using only the relevance ranking. In this way we wanted to validate the usefulness of emotional intensity and specificity rankings in the process. Finally, we considered the performance of our method without application of the rating star.

For the evaluation, we use NDCG (Järvelin and Kekäläinen, 2002) metric. NDCG measures the quality of ranking or recommendation system using list positions. For the purpose of ranking evaluation with NDCG, we use the helpfulness vote’s feature as the relevance value to determine the ranking. The relevance value for the NDCG is calculated based on the helpfulness vote obtained from Amazon using the gold standard as in (Woloszyn et al., 2017). The gold standard formula is in Eq 3:

\[
H(r \in R) = \frac{vote_+(r)}{vote_+(r) + vote_-(r)}
\]

Where \(r\) is a review, \(vote_+\) is the number of customers who voted for the review as being helpful and \(vote_-\) is for the customers who votes it as being unhelpful. \(H(r \in R)\) is then used as the relevance value for the NDCG.

5 Result and Discussion

The results obtained by each of the evaluated methods on each of the four datasets are presented in Tables 2-5.

Each table demonstrates the result obtained by each of the methods with and without incorporating the star rating in the process. The first row in each of the tables refers to the results obtained by the state-of-the-art (MRR) unsupervised baseline (Woloszyn et al., 2017). Rows 2 and 3 show the results obtained by our method based on only the relevance ranking and using the full text or the summary of the reviews, respectively. The last two rows refer to the results obtained by the method when all three rankings were incorporated in the process. We evaluate our ranking quality using NDCG metrics and we take four different ranking positions. Those are NDCG@3, NDCG@5, NDCG@7, and NDCG@10 where the number after the NDCG@ represent the number of reviews taken for evaluation from the top rank position.

From the results presented in Tables 2-5 we can observe that for each of the four datasets, the proposed method performed better when all three ranking were incorporated. This indicated that the emotional intensity and the specificity of a review are useful when determining its helpfulness. It can also be noted that our method obtained better results when the star rating is used when creating the final ranking. The difference is particularly apparent for the Books dataset. Finally, we can see that the proposed method performed better when applied with the full review content (Relevance(full text)+emotion+specify) than with the summary (Relevance(summary)+emotion+specify) with three out of four datasets. The only case when using the summaries of the reviews made a positive difference is the CDs & Vinlys dataset. Looking at the overall results (Both with and without rating star) we can conclude that our proposed method performs best when each of the three rankings is performed on the full reviews’ content and when the rating star is considered.

When comparing the proposed method with the baseline (MRR), we can observe from Tables 2-5 that we obtained better results according to each of the evaluation scores (NDCG@3, NDCG@5, NDCG@7, NDCG@10) in all datasets. For example, Table 2 shows our combination ranking score (relevance+emotion+specify) at NDCG@3, NDCG@5, NDCG@7, NDCG@10 are 0.982, 0.977, 0.974, and 0.972, respectively which improves by 1% from the baseline. On other datasets, the improvement is showing up to 2% compare with the baseline at NDCG@5 on Books dataset and NDCG@3 on Movies & TV dataset. To further evaluate the proposed method in comparison to the baseline, we assess whether the differences in their

\(^4\)https://pypi.org/project/sumy/
performances are statistically significant using the T-test. According to 0.05 significance level, the difference was statistically significant in 11 out of 16 cases. As the 16 cases we consider four different performance measures. (NDCG@3, NDCG@5, NDCG@7, NDCG@10) calculated for each of the four datasets. The results obtained by our method on the books and CDs & Vinyls datasets are numerically superior in all four cases. As for electronic dataset, only NDCG@3 and NDCG@5 are statistically different, while on Movie & TV dataset, only one result that shows the difference in statistic, it is NDCG@10.

### 6 Conclusion

This paper addresses the problem of online reviews ranking according to their helpfulness. We propose an unsupervised method, which first ranks the reviews based on their relevance, emotional intensity and specificity and then combine them in order to obtain the final helpfulness ranking. The performance of the method on four datasets that were created for the purpose of this study was evaluated using the NDCG metric. It was demonstrated that the method outperformed the state-of-the-art unsupervised online review ranking method proposed in (Woloszyn et al., 2017) in every case. In the future, we want to improve our ranking system by applying different features and ranking method. Some features such as linguistic features, positive and negative emotion, or topic sentences may be explore in the ranking system. Moreover, different combination ranking method such as Schulze (Schulze, 2018) or Borda count (Emerson, 2013) or another ranking method could be explored to improve the performance.

### References

Abdalraheem Alsmdai, Shadi Alzu’bi, Bilal Hawashin, Mahmoud Al-Ayyoub, and Yaser Jararweh. 2020. Employing Deep Learning Methods for Predicting Helpful Reviews. 2020 11th International Confer-
Amazon.com. *Amazon’s Top Customer Reviewers*.


Pei Ju Lee, Ya Han Hu, and Kuan Ting Lu.


Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. 2013a. What yelp fake review filter might be doing? In *Seventh international AAAI conference on weblogs and social media*.


Jared Spool. 2009. The Magic Behind Amazon’s 2.7 Billion Dollar Question—UX Articles by UIE.


incom.py 2.0 – Calculating Linguistic Distances and Asymmetries in Auditory Perception of Closely Related Languages

Marius Mosbach¹,², Irina Stenger¹, Tania Avgustinova¹, Bernd Möbius¹, Dietrich Klakow¹,²
¹ Department of Language Science and Technology, Saarland University, Germany
² Saarland Informatics Campus
{mosbach, dlakow}@lsv.uni-saarland.de
ira.stenger@mx.uni-saarland.de
{avgustinova, moebius}@lst.uni-saarland.de

Abstract
We present an extended version of a tool developed for calculating linguistic distances and asymmetries in auditory perception of closely related languages. Along with evaluating the metrics available in the initial version of the tool, we introduce word adaptation entropy as an additional metric of linguistic asymmetry. Potential predictors of speech intelligibility are validated with human performance in spoken cognate recognition experiments for Bulgarian and Russian. Special attention is paid to the possibly different contributions of vowels and consonants in oral intercomprehension. Using incom.py 2.0 it is possible to calculate, visualize, and validate three measurement methods of linguistic distances and asymmetries as well as carrying out regression analyses in speech intelligibility between related languages.

1 Introduction
1.1 Background
Individual (receptive) multilingualism is essential in the European cultural and economic area.¹ How can native speakers of L1 spontaneously understand Lx on the basis of L1-Lx relatedness? Common experience shows that the degree of success in intercomprehension differs between spoken and written modalities, due to various linguistic and non-linguistic factors (Gooskens, 2019). While in listening the time available for the auditory input processing is limited, in reading one can jump back at will (Möller and Zeevaert, 2015) during visual input processing. The latter scenario, however, excludes the possibility to check comprehension through interactive communicative feedback. Even though cross-lingual differences are a multidimensional phenomenon (van Heuven, 2008), a simple baseline prediction of speech intelligibility between related languages can be based on the similarity of their phoneme inventories and lexicons.

For example, both similarity of phoneme inventories and lexical similarity were identified as factors predicting the perceptual confusion ability (and, by implication, similarity) of languages in the Great Language Game (Skirgård et al., 2017). These authors report confusion asymmetries within the set of Slavic languages, i.e., cases of speakers of one language (e.g. Bulgarian) understanding another language (e.g. Russian) better than the other way round. Phonetic and lexical distances are also considered as determinants of mutual intelligibility in the work of Gooskens and colleagues, who use a game-like interface, MICReLa², to collect data for Germanic, Romance, and (six) Slavic languages (Gooskens and van Heuven, 2020).

Notions of entropy and surprisal (Shannon, 1948) are employed in the INCOMSLAV framework³ to measure information density and gauge how language users master high degrees of surprisal due to partial incomprehensibility of linguistic encodings. The key idea is that understanding an unknown but related Lx should be better when the L1 language model adapted for processing the unknown Lx exhibits relatively low average surprisal, or information density (Fischer et al., 2017; Jágrová et al., 2018; Stenger et al., 2017).

¹https://ec.europa.eu/education/policies/multilingualism/about-multilingualism-policy_en
²http://www.micrela.nl/app/
³https://intercomprehension.coli.uni-saarland.de

972
Proceedings of Recent Advances in Natural Language Processing, pages 972–981
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_111
1.2 This Paper

In the present study we extend the incom.py toolbox\(^4\) (Mosbach et al., 2019) focusing on mutual intelligibility aspects in oral comprehension. First, we compare the available measuring methods for linguistic distances and asymmetries – i.e., Levenshtein distance and word adaptation surprisal – as predictors of mutual intelligibility in auditory perception and add word adaptation entropy as an additional metric for asymmetric intelligibility. At the same time, we consider phonetic aspects, in particular vowel and consonant (dis)similarities, as explaining variables in spoken word intelligibility tests. While the initial version of incom.py (Mosbach et al., 2019) provides a baseline to calculate linguistic distances and asymmetries between related languages in visual perception, the current modification, referred to here as incom.py 2.0, targets the mutual intelligibility in auditory perception. The contributions of this paper include:

- Evaluation of two metrics for computing distances and asymmetries based on symbolic phonetic notations, including vowel and consonant (dis)similarities.
- Implementation of word adaptation entropy as an additional predictor of asymmetric intelligibility on the word level.
- Validation of presented predictors and explaining variables in spoken word translation tasks for Bulgarian and Russian.

2 incom.py 2.0

We borrow notation from Mosbach et al. (2019) and let \(L\) denote a language such as Russian or Bulgarian. Each language \(L\) has an associated alphabet – a set of characters – \(A(L)\) which includes the special symbol \(\emptyset\). We use \(w \in L\) to denote a word in language \(L\) and \(c_i \in w\) to denote the \(i\)-th character in word \(w\). Similarly, we will use IPA\((w)\) to denote the symbolic phonetic representation of \(w\), with \(A_{IPA}(L)\) being the phonetic alphabet of \(L\) and \(s_i \in IPA(w)\) denoting the \(i\)-th sound in word \(w\). Moreover, let \(\mathcal{V}(IPA(w))\) denote the set of all vowels in \(w\) and \(C(IPA(w))\) the set of all consonants in \(w\), respectively.

LD and WAS Following Mosbach et al. (2019), incom.py 2.0 supports computing the Levenshtein distance (LD) and word-adaptation surprisal (WAS) as well as their normalized versions \(nLD\) and \(nWAS\) between a pair of words \(w_i, w_j\) on the orthographic level. We extend these to the phonetic level by computing \((n)LD\) and \((n)WAS\) between IPA\((w_i)\) and IPA\((w_j)\), \(\mathcal{V}(IPA(w_i))\) and \(\mathcal{V}(IPA(w_j))\) as well as \(C(IPA(w_i))\) and \(C(IPA(w_j))\). We refer to Mosbach et al. (2019) for the mathematical definitions an in-depth discussion of \((n)LD\) and \((n)WAS\).

Identical Phonetic Correspondences Additionally, incom.py 2.0 supports the computation of the number of identical phonetic correspondences between two words \(w_i\) and \(w_j\) based on their phonetic representations IPA\((w_i)\) and IPA\((w_j)\). We compute the number of identical phonetic correspondences by first applying the LD algorithm of Mosbach et al. (2019) to IPA\((w_i)\) and IPA\((w_j)\) to obtain their alignment and then simply counting the number of identical phonetic transcription agreements.

Word Adaptation Entropy Lastly, incom.py 2.0 supports the computation of the normalized word adaptation entropy (nWAE) between two words \(w_i\) and \(w_j\). Recall that given a character (sound) \(c \in A(L1)\) (\(c \in A_{IPA}(L1)\)) and another character (sound) \(t \in A(L2)\) (\(t \in A_{IPA}(L2)\)), the character (sound) adaptation surprisal between \(s\) and \(t\) is defined as follows: \(\text{CAS}(s, t) = -\log_2(P(t|s))\). Now, nWAE between two words \(w_i\) and \(w_j\) can be computed by first obtaining their aligned sequences \(\tilde{w}_i, \tilde{w}_j\) using the LD algorithm of Mosbach et al. (2019) followed by computing the average (phonetic) character entropy as follows:

\[
\text{nWAE}(\tilde{w}_i, \tilde{w}_j) = \frac{1}{L} \sum_{l=0}^{L-1} P(t_l|s_l) \times \text{CAS}(s_l, t_l)
\]

3 Experimental Setup

Via the INCOMSLAV platform\(^5\), we conducted a series of mutual intelligibility tests in the spo-

---

\(^4\)Our code is available online: \url{https://github.com/uds-lsv/incompy}

\(^5\)The website includes a large number of different online experiments in 11 Slavic languages (as well as
ken modality with isolated cognate (i.e., historically or etymologically related) words. Being a challenging task on its own, the context-free recognition of cross-lingual cognates is a precondition of successful oral intercomprehension (Gooskens and van Heuven, 2020). Here, we investigate the transparency of Bulgarian–Russian cognates with regard to diachronically motivated sound correspondences that appear to facilitate or hinder human performance. As we are interested in inherent intercomprehension, only people who speak their first language (L1) natively and who do not know the stimulus language (Lx) have been included in the analysis. After completing a background questionnaire, Bulgarian (BG) and Russian (RU) participants were asked to translate randomized 1206 oral7 BG and RU stimuli into their respective native language in two series of 60 stimuli each. The items were taken from the written intelligibility tests of Mosbach et al. (2019) in order to obtain a reliable baseline.

The number of RU participants is 29, aged between 16 and 48 years (average age 32)8 with 23 women, 5 men, and 1 not specified. The number of BG participants is 11, aged between 19 and 37 years (average age 27)9 with 10 females and 1 male. Even though these two groups differ considerably in size, statistical analyses based on fewer participants are particularly worthwhile for the practice of foreign language learning and teaching and in experiments involving specific target groups (Branets et al., 2020). In this study, when no statistical observations can be made, we report findings as percentages to indicate success rates based on collected material, bearing in mind that our results should be interpreted with caution as general trends.

During the experiments the participants listened to the stimuli one by one (each word was played twice), and had to provide a written translation within 10 seconds. The order of stimuli was randomized. The time limit was chosen based on the experience from other intercomprehension experiments (Golubović, 2016), and the results were automatically categorized as ‘correct’ or ‘wrong’ via pattern matching with pre-defined correct answers and acceptable alternatives. The responses were then manually checked for typographical errors in the final analysis. The mean percentage of correctly translated items constitutes the intercomprehension score in each language (Table 1). These results show that the RU participants understand BG words at 68.42% and that the BG participants understand RU cognates they are presented with at 65.58%. This suggests that the intelligibility of spoken BG and RU stimuli did not cause major problems for the respective native speakers. For comparison, the intelligibility scores presented in the study of Mosbach et al. (2019) for the written modality are slightly higher, e.g. 71.33% (BG for RU) respectively 74.67% (RU for BG).

## 4 Results

### 4.1 Available Measures

The statistical analyses in Mosbach et al. (2019) clearly supported the normalized Levenshtein distance (nLD) as a reliable predictor of orthographic intelligibility on the word level for BG and RU, while the predictive potential of normalized word adaptation surprisal (nWAS) was rather weak despite its modification. From a cross-linguistic perspective, the advantage of surprisal-based methods, in contrast to Levenshtein distance, is in capturing asymmetries (Stenger et al., 2020). In this study, we validate the correlation of nLD and nWAS with the intelligibility scores obtained in oral intercomprehension experiments.

<table>
<thead>
<tr>
<th>L1</th>
<th>Spoken stimuli</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulgarian</td>
<td>65.58%</td>
</tr>
<tr>
<td>Russian</td>
<td>68.42%</td>
</tr>
</tbody>
</table>

Table 1: Intelligibility scores from free translation tasks by humans in auditory perception.

---

6In German and English) carried out as challenges in a linguistic game (https://intercomprehension.coli.uni-saarland.de); for more details about intercomprehension experiments see Stenger et al. (2020).
7118 nouns and 2 numerals in each language.
8BG and RU stimuli were read aloud by Bulgarian and Russian female native speakers and recorded in a professional sound studio.
9In total 30 participants took part at the experiments, one male participant noticed the knowledge of Bulgarian and was excluded from the analysis.
10In total 13 participants took part at the experiments, two participants noticed the knowledge of Russian and were excluded from the analysis.
Levenshtein Distance. As the simplest version of the Levenshtein algorithm is based on binary differences between alignments, we use a modified version with vowels matched only to vowels and consonants only to consonants. However, it is not always clear how to assign the respective weights in modeling speech intelligibility (Gooskens, 2019). The optimal weighting differs for each language combination, taking into consideration the predictability and generalizability of sound correspondences as well as the human decoding process (Berthele, 2011; Gooskens et al., 2008; Kürschner et al., 2008; Möller and Zeevaert, 2015; Stenger and Avgustinova, 2020). The version that we use in this article is based on a cost matrix including communicatively relevant sound distances for speech intelligibility between BG and RU (Stenger and Avgustinova, 2020).

To reveal the relationship between speech intelligibility and nLD, we correlated the results of the spoken cognate recognition tests with the calculated nLD in each language group. The basic assumption is that small distances would correlate with high intelligibility scores, while large distances are expected to correlate with low intelligibility scores. Figure 1 shows a significant negative correlation in both directions: BG for RU: Pearson’s $r = -0.25$, $p = 0.0056$ and RU for BG: Pearson’s $r = -0.4$, $p = 5.7\times10^{-6}$. This means that cross-lingual spoken word intelligibility is predictable from the nLD between the stimulus and the native language. The larger the distances the more difficult it is to recognize the cognates of a related language. However, the nLD accounts for 16.08% ($R^2 = .16075$) of the variance in the intelligibility scores for BG listeners and for only 6.33% ($R^2 = .06325$) of the variance in the intelligibility scores for RU listeners, leaving the majority of the variance unexplained. Hence, for RU, the predictive potential of nLD in the oral modality is smaller than in the written one in Mosbach et al. (2019).

Word Adaptation Surprisal. We correlated the nWAS values with the intelligibility scores to find out whether surprisal can predict human performance in spoken cognate recognition. With smaller nWAS values we expect an easier cognate guessing effect. The correlations between nWAS and the intelligibility scores are shown in Figure 2. We found negative correlations between nWAS and the intelligibility scores for both written and spoken modalities. Additionally, we investigated to what extent (dis)similarities in vowels and consonants may contribute to the cognate recognition process between BG and RU. As Gooskens and van Heuven (2020, p. 376) point out, consonants are better predictors of speech intelligibility.
than vowels, e.g., in the intelligibility of Scandinavian dialects for speakers of Standard Danish, consonants are more important than vowels (Gooskens et al., 2008; van Heuven, 2008).

We calculated the normalized vowel and consonant distances (nLD; see Section 2) and correlated these metrics with the intelligibility scores (Figure 3). The negative correlations for the vowels were significantly stronger than for consonants: Pearson’s $r = -.222$ ($p < .05$) versus Pearson’s $r = -.127$ ($p = .167$) for BG–RU language pair and Pearson’s $r = -.390$ ($p = 1.04e-05$) versus Pearson’s $r = .019$ ($p = .835$) for RU–BG language pair. Although the correlations are lower than in the case of the full nLD (considering all classes of speech sounds), the vowel distances are significant in both language groups. The BG–RU consonant distance is still negative but not significant. In the opposite direction, the RU–BG consonant distance is not negative as assumed but again not significant. It might be the case that qualitative characteristics of the mismatched consonant correspondences in spoken word recognition are more decisive than their number. In spoken cognate recognition between BG and RU, vowels turn out to be empirically more important than consonants. The impact of the total number of vowels (V) and consonants (C) in the stimulus word and of the identical vowel (Vic) and consonant (Cic) correspondences between the stimulus and the target cognates in spoken word recognition are correlated as explaining variables with the obtained intelligibility scores. Table 2 provides an overview of the statistical results (Pearson’s r and p-value).

The statistical results show significantly positive correlations between the number of vowels and consonants in the stimulus cognates and the intelligibility score of the BG–RU language pair. The positive correlation for consonants is, in this case, significantly stronger than for vowels. However, the correlation for the identical vowel correspondences is significantly stronger, while the correlation for the identical consonant correspondences is positive, but insignificant.

For the RU–BG language pair, statistical results show significantly positive correlations of intelligibility scores with the number of consonants in the stimulus cognates and with identical vowel correspondences. The predictive potential of the number of vowels in auditory perception of RU cognates by BG participants is, however, small and insignificant. The correlation for identical consonant correspondences is not positive as assumed but again insignificant.

How to interpret these results? First of all, in our experimental material, the cognate BG words are slightly shorter than their RU equivalents (4.64 vs. 4.81). Looking at BG and RU sound inventories in more detail, we see an almost equal distribution of consonants between the two languages: 343 vs. 341 (2.86 per word in BG and 2.84 per word in RU). Consonants of the two languages include 224 identical correspondences (38.89%) and 122 mismatched correspondences (21.18%), including missing consonants. At the same time, the distribution of vowels is not so equally represented between BG and RU: 214 vs. 230 (1.78 per word in BG and 1.92 per word in RU), where vowels of the two languages represent only 53 identical correspondences (9.20%) and
177 mismatched correspondences (30.73%), including missing vowels. These facts partly explain the results obtained from correlations and the different contribution of vowels and consonants to understanding spoken cognates.

To conclude, the experimental subjects seemed to rely on the interplay between a vowel (dis)similarity effect and a larger number of consonants with a balanced distribution. On the other hand, it might be the case that participants are communicatively more or less tolerant (or sensible) to different phonetic details (Gooskens et al., 2015), in particular to the qualitative effect of sound correspondences in spoken cognate recognition (Stenger and Avgustinova, 2020).

### 4.3 Word Adaptation Entropy

Despite all dissimilarities, speech intelligibility between related languages appears to be possible mainly because of regular correspondences. Speakers of L1 can understand a word \( w_j \) from a related Lx insofar as they can predict which word \( w_i \) of L1 is the best equivalent for \( w_j \).

In the case of cognates, the prediction can be based on sound correspondences. Entropy, as the mean of surprisal, gives a quantification of the overall uncertainty involved in making a choice. Let us consider an example. From a RU perspective, the BG–RU cognate pair [dup]-[dup] “oak” gives us individual sound adaptation entropies of 0.567 for [d] (BG [d] potentially corresponds to RU [d] and [d’]), 1.689 for [v] (BG [v] potentially corresponds to RU [u], [e], [o] and [i]) and 0.523 for [p] (BG [p] potentially corresponds to RU [p] and [p’]). An aggregate measure for the entire word thus results in 2.778, which after normalization by the sound correspondence alignment gives us 0.926. From a BG perspective, on the other hand, the RU–BG cognate pair [dyp]-[dup] “oak” gives us individual sound adaptation entropies of 0.0 for [d] (RU [d] can only correspond to BG [d]) as well as for [p] (RU [p] can only correspond to BG [p]), but 1.296 for [u] (RU [u] potentially corresponds to BG [v], [a] and [u]). As an aggregate measure for the entire word we get 1.296 and eventually normalized value of 0.432.

Based on sound entropies we have calculated the normalized word adaptation entropy (nWAE) as an aggregate measure for entire words. We assume that the smaller the nWAE, the easier it is to guess the spoken cognate in a related language. We found negative but very low and not significant correlations in both directions: BG for RU: Pearson’s \( r = -0.078, p = .40 \) and RU for BG: Pearson’s \( r = -0.069, p = .46 \). The question is why the correlations are so low and not significant? Our analysis is based on a limited number of stimulus words (120 cognates in each direction) and this sample may be too limited for nWAE values (Moberg et al., 2006). Additional linguistic factors may influence the mutual intelligibility of related languages, e.g., the existence of
words that are very similar to the stimulus but differ only in one sound, which leads to disregarding the correct counterpart (Kürschner et al., 2008). We therefore assume that WAE needs to systematically account for such neighborhood density effects as a relevant factor for speech intelligibility.

By using the nWAE values, we can now quantify the uncertainty in the overall adaptation process of cognates. For BG and RU, we found that the mean nWAE for RU given BG (1.267) is higher than for BG given RU (0.807). This means that a RU listener may in general have more difficulties when exposed to BG than a BG speaker exposed to RU. There are 114 BG cognates with higher nWAE values in comparison to 114 RU cognates. This means that RU subjects listening to 114 BG stimuli cognates are predicted to deal with more uncertainty than BG subjects listening to 114 RU stimuli cognates. Only 6 RU cognates have higher nWAE values in comparison to BG cognates. In this case BG subjects adapting 6 RU stimuli cognates are predicted to face larger uncertainty than RU subjects trying to adapt 6 BG stimulus cognates.

To carry out comparative analyses with the small group of BG participants, we automatically extracted a corresponding sample of 11 RU subjects with matching intelligibility scores. We then calculate the difference in intelligibility between the two groups separately for all cognate pairs. The quantitative data show that 56 BG stimuli have higher intelligibility scores than their RU cognates, 50 RU stimuli have higher intelligibility scores than their BG cognates, and 14 cognate pairs have identical scores. Let us consider six cognate pairs with higher nWAE values for the stimuli, which imply larger difficulties for the listeners. There are indeed three BG stimuli with higher intelligibility scores than their RU cognates, cf. BG–RU [3gon]–[3gon] “flame/fire” (91% vs. 29%), [3ol]–[3ol] “salt” (100% vs. 90%) and [3lo][3a]–[3lo][3a] “knee” (45% vs. 29%). On the other hand, three RU stimuli have higher intelligibility scores than their BG cognates despite higher nWAS values, cf. RU–BG [4gerox]–[4grax] “pea” (50% vs. 27%), [4mto]–[4jatsc] “egg” (86% vs. 82%) and [4porox]–[4prax] ”gunpowder” (14.0% vs. 9.0%).

To recap, the obtained experimental results indicate that the calculated nWAE values can only partially explain the asymmetric intelligibility of spoken words between the two tested languages.

4.4 Regression Analyses

In this Section we present regression analysis results. The nLD correlates most strongly with intelligibility scores in both directions (see Section 4.1). The negative correlation between the intelligibility score and the nWAS is very small and significant only for BG listeners (see Section 4.1). However, in order to investigate whether the nWAS still has a significant additional contribution to speech intelligibility in case of BG and RU, a multiple regression analysis was performed. Note that even though the nWAE does not correlate significantly with the intelligibility scores in both directions (see Section 4.3) it is included as the third predictor in the regression analysis. Figure 4 and Figure 5 present the results of the regression analysis, conducted first with the method to identify the effect of two predictors in combination: nLD and nWAS, nLD and nWAE, and nWAS and nWAE.

The first regression model containing nLD and nWAS as two predictors could account for 16.12% ($R^2 = .16117, p = 3.421e–05$) of the variance in intelligibility of spoken RU words and for only 6.37% ($R^2 = .06370, p < .05$) of the variance in intelligibility of spoken BG words. The second model containing nLD and nWAE as two predictors explains the variance to a slightly better extent: RU for BG – 16.30% of the variance ($R^2 = .16296, p = 0.20338e–05$) and BG for RU – 6.59% ($R^2 = .06592, p < .05$). However, the third model of nWAS and nWAE does not explain the variance to a significant extent in both directions: RU for BG – 4.00% of the variance ($R^2 = .03999, p = .092$) and BG for RU – 1.64% of the variance ($R^2 = .01637, p = .381$). The final model includes all three predictors: nLD, nWAS and nWAE and this results in explaining 16.74% ($R^2 = .16742, p = 8.9276e–05$) of the variance in understanding of RU spoken cognates and in only 6.60% ($R^2 = .06600, p < .05$) of the variance in understanding of BG spoken cognates. A combination of nLD, nWAS and nWAE is only a slightly better predictor of speech intelligibility in case
of BG and RU than two predictors: nLD and nWAS or nLD and nWAE, and than nLD alone. Nevertheless, the amount of explained variance is low for all three predictors.

5 Conclusion and Outlook

We presented incom.py 2.0, the extended version of the incom.py toolkit for calculating similarities and asymmetries between closely related languages, with a focus on auditory perception. Our main conclusions are the following:

- Our statistical analyses clearly support nLD as a reliable predictor of speech intelligibility on the word level, although its predictive potential in the BG–RU setup is smaller for the oral modality when compared to the written one (Mosbach et al., 2019). The predictive potential of nWAS for speech intelligibility in case of BG and RU is rather weak and significant only for BG speakers, which is in line with the results for the written modality (Mosbach et al., 2019).

- Our special attention to the possibly different contributions of vowels and consonants in oral intercomprehension has shown that in the experimental material vowels are more important than consonants in recognizing cognates between BG and RU. A closer look at sound correspondences further reveals that a larger number of consonants with a balanced distribution between the two languages also plays a role in spoken word recognition.

- The additional linguistic asymmetry factor of nWAE quantifies the overall uncertainty involved in making a choice in spoken word recognition. We correlated nWAE with intelligibility scores and found negative correlations between the nWAE values and the intelligibility scores in both directions. However, the correlations are very low and not significant. Hence, nWAE can only partly explain or predict asymmetric intelligibility of spoken words between the two tested languages, which we attribute to the small empirical base of the present study.

- We carried out regression analyses to find out which combination of presented predictors served better to explain speech intelligibility in our experiments. We found out that the combination of nLD, nWAS and nWAE is a slightly better predictor of speech intelligibility in case of BG and RU than two predictors nLD and nWAS or nLD and nWAE and than nLD alone. Nevertheless, the amount of explained variance is low for three predictors.

As a next step, we plan to extend the range of the studied phenomena beyond mere word similarity and to include predictability of constructional similarity by focusing on cross-lingual correspondences of multi-component units in phrasal and sentential contexts. Additionally, we plan to integrate reaction times into the experiments in order to study the interaction of linguistic and cognitive aspects of human performance.
Acknowledgments
This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 232722074 – SFB 1102.

References


Appendix
Figure 5 shows the relationship between speech intelligibility and the various predictors introduced in Section 2 in the direction BG for RU.
Figure 5: Relationship between speech intelligibility and nLD and nWAS (5a), nLD and nWAE (5b), and nWAS and nWAE (5c) in the direction BG for RU.
Not All Linearizations Are Equally Data-Hungry in Sequence Labeling Parsing

Alberto Muñoz-Ortiz¹, Michalina Strzyz¹,², David Vilares¹
¹Universidade da Coruña, CITIC, Spain
²Priberam Labs, Portugal
alberto.munoz.ortiz@udc.es
michalina.strzyz@priberam.pt, david.vilares@udc.es

Abstract

Different linearizations have been proposed to cast dependency parsing as sequence labeling and solve the task as: (i) a head selection problem, (ii) finding a representation of the token arcs as bracket strings, or (iii) associating partial transition sequences of a transition-based parser to words. Yet, there is little understanding about how these linearizations behave in low-resource setups. Here, we first study their data efficiency, simulating data-restricted setups from a diverse set of rich-resource treebanks. Second, we test whether such differences manifest in truly low-resource setups. The results show that head selection encodings are more data-efficient and perform better in an ideal (gold) framework, but that such advantage greatly vanishes in favor of bracketing formats when the running setup resembles a real-world low-resource configuration.

1 Introduction

Dependency parsing (Mel’cuk et al., 1988; Kühbler et al., 2009) has achieved clear improvements in recent years, to the point that graph-based (Martins et al., 2013; Dozat et al., 2017) and transition-based (Ma et al., 2018; Fernández-González and Gómez-Rodríguez, 2019) parsers are already very accurate on certain setups, such as English news. In this line, Berzak et al. (2016) have pointed out that the performance on these setups is already on par with that expected from experienced human annotators.

Thus, the efforts have started to focus on related problems such as parsing different domains or multi-lingual scenarios (Sato et al., 2017; Song et al., 2019; Ammar et al., 2016), creating faster models (Volokh, 2013; Chen and Manning, 2014), designing low-resource and cross-lingual parsing techniques (Tiedemann et al., 2014; Zhang et al., 2019), or infusing syntactic knowledge into models (Strubell et al., 2018; Rotman and Reichart, 2019).

This work will lie in the intersection between fast parsing and low-resource languages. Recent work has proposed encodings to cast parsing as sequence labeling (Spoustová and Spousta, 2010; Strzyz et al., 2019; Gómez-Rodríguez et al., 2020; Li et al., 2018; Kiperwasser and Ballesteros, 2018). This approach computes a linearized tree of a sentence of length \( n \) in \( n \) tagging actions, providing a good speed/accuracy trade-off. Also, it offers a naïve way to infuse syntactic information as an embedding or feature (Ma et al., 2019; Wang et al., 2019). Such encodings have been evaluated on English and multi-lingual setups, but there is no study about their behaviour on low-resource setups, and what strengths and weaknesses they might exhibit.

Contribution We study the behaviour of linearizations for dependency parsing as sequence labeling in low-resource setups. First, we explore their data efficiency, i.e. if they can exploit their full potential with less supervised data. To do so, we simulate different data-restricted setups from a diverse set of rich-resource treebanks. Second, we shed light about their performance on truly low-resource treebanks. The goal is to determine whether tendencies from the experiments in the previous phase hold when the language is truly low-resource and when secondary effects of real-world low-resource setups, such as using predicted part-of-speech (PoS) tags or no PoS tags, impact more certain types of linearizations.

2 Related work

Low-resource parsing has been explored from perspectives such as unsupervised parsing, data augmentation, cross-lingual learning, or data-efficiency of models. For instance, on unsupervised parsing, Klein and Manning (2004) and Spitkovsky et al. (2010) have worked on generative models to determine whether to continue or stop attaching de-
pendents to a token, while others (Le and Zuidema, 2015; Mohananey et al., 2020) have studied how to use self-training for unsupervised parsing.

On data augmentation, McClosky et al. (2006) used self-training to annotate extra data, while others have focused on linguistically motivated approaches to augment treebanks. This is the case of Vania et al. (2019) or Dehouck and Gómez-Rodríguez (2020), who have proposed methods to replace subtrees within a given sentence.

On cross-lingual learning, authors such as Søgaard (2011) or McDonald et al. (2011) trained delexicalized parsers in a source rich-resource treebank, which are then used to parse a low-resource target language. Falenska and Çetinoğlu (2017) explored lexicalized versus delexicalized parsers and compared them on low-resource treebanks, depending on factors such as the treebank size and the PoS tags performance. Wang and Eisner (2018) created synthetic treebanks that resemble the target language by permuting constituents of distant treebanks. Naseem et al. (2012) and Täckström et al. (2013) tackled this same issue, but from the model side, training on rich-resource languages in such a way the model learns to detect the aspects of the source languages that are relevant for the target language. Recently, Mulcaire et al. (2019) used an LSTM to build a polyglot language model, which is then used to train on top of it a parser that shows cross-lingual abilities in zero-shot setups.

On data-efficiency, research work has explored the impact of the use of different amounts of data, motivated by the lack of annotated data or by the lack of quality of it. For instance, Lacroix et al. (2016a) showed how a transition-based parser with a dynamic oracle can be used without any modifications to parse partially annotated data. They found that this setup is useful to train low-resource parsers on sentence-aligned texts, from a rich-resource treebank to an automatically translated low-resource language, where only precisely aligned tokens are used for the projection in the target dataset. Lacroix et al. (2016b) studied the effect that pre-processing and post-processing has in annotation projection, and concluded that quality should prevail over quantity. Related to training with restricted data, Anderson and Gómez-Rodríguez (2020) showed that when distilling a graph-based parser for faster inference time, models with smaller treebanks suffered less. Dehouck et al. (2020) also distilled models for Enhanced Universal Dependencies (EUD) parsing with different amounts of data, observing that less training data usually translated into slightly lower performance, while offering better energy consumption. Garcia et al. (2018) showed, in the context of Romance languages, that peeking samples from related languages and adapting them to the target language is useful to train a model that performs on par with one trained on fully (but still limited) manually annotated data. Restricted to constituent parsing, Shi et al. (2020) analyzed the role of the dev data in unsupervised parsing. They pointed out that many unsupervised parsers use the score on the dev set as a signal for hyper-parameter updates, and show that by using a handful of samples from that development set to train a counterpart supervised model, the results outperformed those of the unsupervised setup. Finally, there is work describing the impact that the size of the parsing training data has on downstream tasks that use syntactic information as part of the input (Sagae et al., 2008; Gómez-Rodríguez et al., 2019).

3 Preliminaries

In what follows, we review the existing families of encodings for parsing as sequence labeling (§3.1) and the models that we will be using (§3.2).

3.1 Encodings for sequence labeling dependency parsing

Sequence labeling assigns one output label to every input token. Many problems are cast as sequence labeling due to its fast and simple nature, like PoS tagging, chunking, super tagging, named-entity recognition, semantic role labeling, and parsing. For dependency parsing, to create a linearized tree it suffices to assign each word $w_i$ a discrete label of the form $(x_i, l_i)$, where $l_i$ is the dependency type and $x_i$ encodes a subset of the arcs of the tree related to such word. Although only labels seen in the training data can be predicted, Strzyz et al. (2020) show that the coverage is almost complete. We distinguish three families of encodings, which we now review (see also Figure 1).

Head-selection encodings (Spoustová and Spousta, 2010; Li et al., 2018; Strzyz et al., 2019). Each word label component $x_i$ encodes its head as an index or an (abstracted) offset. This can be done by labeling the target word with the (absolute) index of its head token, or by using a relative offset that accounts for the difference between the dependent and head indexes. In this work, we
chose a relative PoS-based encoding (rpxb) that has shown to perform consistently better among the linearizations of this family. Here, each \( x_i \) is a tuple \((p_i, o_i)\), such that if \( o_i > 0 \) the head of \( w_i \) is the \( o_i \)th word to the right of \( w_i \) whose PoS tag is \( p_i \); if \( o_i < 0 \), the head of \( w_i \) is the \( o_i \)th word to the left of \( w_i \) whose PoS tag is \( p_i \). Among its advantages, we find the capacity to encode any non-projective tree and words being directly and only linked to its head, but on the other hand it is dependent on external factors (e.g. PoS tags).\(^1\)

**Bracketing-based encodings** (Yli-Jyrä and Gómez-Rodríguez, 2017). Each \( x_i \) encodes a sort of incoming and outgoing arcs of a given word and its neighbors, represented as bracket strings. More particularly, in Strzysz et al. (2019) each \( x_i \) is a string that follows the expression \((< \) ? (\( (\) \) x \( (\) \) x)) \( (> \) ?), where \( < \) means that \( w_{i-1} \) has an incoming arc from the right, \( k \) times \( \backslash \) means that \( w_i \) has \( k \) outgoing arcs towards the left, \( k \) times \( / \) means that \( w_{i-1} \) has \( k \) outgoing arcs to the right, and \( > \) means that \( w_i \) has an arc coming from the left. This encoding produces a compressed label set while not relying on external features, such as PoS tags. However, when it comes to non-projectivity, it can only analyze crossing arcs in opposite directions. To counteract this, it is possible to define a linearization using a second independent pair of brackets (denoted with \(*\)) to encode a 2-planar tree (Strzysz et al., 2020).\(^2\)

In this work we are considering experiments with both the restricted non-projective \((rxxb)\) and the 2-planar bracketing encodings \((2pb)\).

**Transition-based encodings** (Gómez-Rodríguez et al., 2020). Each \( x_i \) encodes a sub-sequence of the transitions to be generated by a left-to-right transition-based parser. Given a sequence of transitions \( t = t_1, \ldots , t_m \) with exactly \( n \) read transitions\(^3\), it splits \( t \) into \( n \) chunks and assigns the \( i \)th chunk to the \( i \)th word. Its main advantage is more abstract, allowing to automatically derive encodings relying on any left-to-right transition based parser (including dependency, constituency and semantic parsers). According to Gómez-Rodríguez et al. (2020), they produce worse results than the bracketing encodings, but we include them in this work for completeness. In particular, we consider mappings from arc-hybrid (Kuhlmann et al., 2011) \((ahxb)\) and Covington (2001) \((ctb)\), which are projective and non-projective transition-based algorithms.

To post-process corrupted predicted labels, we follow the heuristics described in each encoding paper.

### 3.2 Sequence labeling framework

**Notes** Let \( w \) be a sequence of words \([w_1, w_2, \ldots , w_{|w|}]\), then \( \vec{w} \) is a sequence of word vectors that will be used as the input to our models. Each \( \vec{w}_i \) will be a concatenation of: (i) a word embedding, (ii) a second word embedding computed through a char-LSTM, (iii) and optionally a PoS tag embedding (we will discuss more about this last point in §4).

We use bidirectional long short-term memory networks (biLSTMs; Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) to train our sequence labeling parsers. BiLSTMs are a strong baseline used in recent work across a number of tasks (Yang and Zhang, 2018; Reimers and Gurevych, 2017). More particularly, we use two layers of biLSTMs, and each hidden vector \( \vec{h}_i \) from the last biLSTM layer (associated to each input vector \( \vec{w}_i \)) is fed to separate feed-forward networks that are in charge of predicting each of the label components of the linearization (i.e. \( x_i \) and \( l_i \)) using soft-maxes, relying on hard-sharing multi-task learning (MTL; Caruana, 1997; Ruder, 2017). Following

---

\(^1\)Other head-selection variants encode arcs based on word properties different than PoS tags (Lacroix, 2019).

\(^2\)An \( x \)-planar tree can be separated into \( x \) planes, where the arcs belonging to the same plane do not cross.

\(^3\)In left-to-right parsers, a read transition is an action that puts a word from the buffer into the stack. For algorithms such as the arc-standard or arc-hybrid this is only the shift action, while in the arc-eager both the shift and right-arc actions are read transitions. See also (Nivre, 2008).
§3.1, for all the encodings, except the 2-planar encoding, we will use a 2-task MTL setup: one task will predict $x_i$ according to each encoding specifics, and the other one will predict the dependency type, $l_i$. For the 2-planar bracketing encoding, which uses a second pair of brackets to predict the arcs from the second plane, we use instead a 3-task MTL setup, where the difference is that the prediction of $x_i$ is split into two tasks: one that predicts the first plane brackets and another task that predicts the brackets from the second plane.\(^4\)

It is worth noting that for this particular work we skipped computational expensive models, such as BERT (Devlin et al., 2019). There are three main reasons for this. First, the experiments in this paper imply training a total of 760 parsing models (see more details in §4), making the training on BERT (or variants) less practical. Second, there is not a multilingual or specific-language BERT model for all languages, and this could be the source of uncontrolled variables that could have an impact on the performance, and thereof on the conclusions.\(^5\)

Third, even under the assumption of all language-specific BERT models being available, these are pre-trained on different data that add extra noise, which could be undesirable for our purpose.

4 Methodology and experiments

We design two studies, detailed in §4.1 and 4.2:

1. We explore if some encodings are more data-efficient than others. To do so, we will simulate data-restricted setups, selecting rich-resource languages and using partial data. The goal is to test if some encodings are learnable with fewer data, or if other ones could obtain a better performance instead, but only under the assumption of very large data being available.

2. We focus on truly low-resource setups. This can be seen as a confirmation experiment to see if the findings under data-restricted setups hold for under-studied languages, and to confirm what sequence labeling linearizations are more recommendable under these conditions.

Experimental setups For experiments 1 and 2, we consider three setups that might have a different impact across the encodings:

1. Gold PoS tags setup: We train and run the models under an ideal framework that uses gold PoS tags as part of the input. The reason is that encodings such as $r^h_p$ rely on PoS tags to rebuild the linearized tree. This way, using gold PoS tags helps estimate the optimal data-efficiency and learnability of these parsers under perfect (but unreal) conditions.

2. Predicted PoS tags setup: Setup 1 cannot truly reflect the performance that the encodings would obtain under real-world data-restricted conditions. Predicted PoS tags will be less helpful because their quality will degrade. This issue can affect more to the $r^h_p$ encoding, since it requires them to rebuild the tree from the labels, and miss-predicted PoS tags could propagate errors during decoding. Here, we train taggers for each treebank, using the same architecture used for the parsers. To be coherent with the data-restricted setups, taggers will be trained on the same amount of data used for the parsers. Appendix A discusses the PoS taggers performance.

3. No PoS tags setup: We train the models without using any PoS tags as part of the input. It is worth noting that the setup is somewhat forced for the $r^h_p$ encoding, since we will still need to externally run the taggers to obtain the PoS tags and rebuild the tree. Yet, we include the PoS-based encoding for completeness, and to have a better understanding about how different families of encodings suffer from not (or minimally) using PoS tags. For instance, that is a simple way to obtain simpler and faster parsing models, as part of the pipeline does not need to be executed, and the input vectors to the models will be smaller, translating into faster executions too. Also, in low-resource setups, PoS tags might not be available or the tagging models are not accurate enough to help deep learning models (Zhou et al., 2020; Anderson and Gómez-Rodríguez, 2021).

4.1 Experiment 1: Encodings data-efficiency

Data We chose 11 treebanks from UD2.7 (Zeman et al., 2020) with more than 10000 training sentences: German\(_{HDT}\), Czech\(_{PDT}\), German\(_{HDT}\), Czech\(_{PDT}\),
Russian SynTagRus, Classical Chinese Kyoto, Persian PDT, Estonian EDT, Romanian Nonstandard, Korean Kaist, Ancient Greek PROIEL, Hindi HDTB and Latvian LVTB. They consider different families, scripts and levels of non-projectivity (see Appendix B). To simulate data-restricted setups, we created training subsets of 100, 500, 1 000, 5 000 and 10 000 samples, as well as the total training set. The training sets were shuffled before the division.

Setup To assess the data-efficiency, we proceed as follows. As the $\text{rp}^h$ encoding has showed the strongest performance in previous work for multi-lingual setups (Strzy et al., 2019; Gómez-Rodríguez et al., 2020), we are taking these models as the reference and an a priori upper bound. Then, we compute the difference of the mean UAS (across the 11 treebanks) between the $\text{rp}^h$ and each of the other linearizations, for all the models trained up to 10 000 sentences. The goal is to determine which encodings suffer more when training with limited data and monitor to what extent the tendency holds as more data is introduced. We compute the statistically significant difference between the $\text{rp}^h$ and the other encodings, using the p-value ($p < 0.05$) of a paired t-test on the scores distribution, following recommended practices for dependency parsing (Dror et al., 2018). Finally, we show specific results for the models trained on the whole treebanks. In this work, we will report UAS over LAS, since the differences in the encodings lie in how they encode the dependency arcs and not their types.

Results Tables 1, 2 and 3 show the difference of the mean UAS for each encoding with respect to the $\text{rp}^h$ one; for the gold PoS tags, predicted PoS tags and no PoS tags setups, respectively. For the gold PoS tags setup, the $\text{rp}^h$ encoding performs better than the bracketing ($\text{rx}^b$ and $\text{2p}^b$) and the transition-based ($\text{ah}^\text{tb}$ and $\text{ct}^\text{tb}$) encodings, for all the training splits. Yet, the gap narrows as the number of training sentence increases. For the predicted PoS tags setup, the relative PoS-based encoding performs better for the smallest set of 100 sentences, but slightly worse for the sets of 500 and 1 000 sentences with respect to $\text{rx}^b$ and $\text{2p}^b$. With more data, the tendency resembles the one from the gold PoS tags setup. Third, for the setup without PoS tags, the tendency reverses. The bracketing encodings perform better, particularly for the smallest test sets, but the gap narrows as the number of training sentences increases.

Table 1: Average UAS difference for the subsets of the rich-resource treebanks under the gold PoS tags setup. Blue and yellow cells show the UAS increase and decrease with respect to the $\text{rp}^h$ encoding, respectively.

<table>
<thead>
<tr>
<th># Sentences</th>
<th>$\text{rp}^h \times$</th>
<th>$\text{rx}^b \times$</th>
<th>$\text{2p}^b \times$</th>
<th>$\text{ah}^\text{tb} \times$</th>
<th>$\text{ct}^\text{tb} \times$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>68.34</td>
<td>-2.15</td>
<td>-2.42</td>
<td>-5.82</td>
<td>-9.96</td>
</tr>
<tr>
<td>500</td>
<td>76.94</td>
<td>-1.58</td>
<td>-1.5</td>
<td>-5.21</td>
<td>-9.35</td>
</tr>
<tr>
<td>1 000</td>
<td>80.29</td>
<td>-1.42</td>
<td>-1.46</td>
<td>-5.16</td>
<td>-8.08</td>
</tr>
<tr>
<td>5 000</td>
<td>86.54</td>
<td>-1.16</td>
<td>-1.26</td>
<td>-3.62</td>
<td>-7.05</td>
</tr>
<tr>
<td>10 000</td>
<td>88.26</td>
<td>-0.8</td>
<td>-0.72</td>
<td>-3.52</td>
<td>-5.67</td>
</tr>
</tbody>
</table>

Table 2: Average UAS difference for the subsets of the rich-resource treebanks under the predicted PoS tags setup.

<table>
<thead>
<tr>
<th># Sentences</th>
<th>$\text{rp}^h \times$</th>
<th>$\text{rx}^b \times$</th>
<th>$\text{2p}^b \times$</th>
<th>$\text{ah}^\text{tb} \times$</th>
<th>$\text{ct}^\text{tb} \times$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>41.87</td>
<td>-0.42</td>
<td>-0.19</td>
<td>-1.9</td>
<td>-3.59</td>
</tr>
<tr>
<td>500</td>
<td>63.45</td>
<td>-0.01</td>
<td>0.14</td>
<td>-1.96</td>
<td>-5.73</td>
</tr>
<tr>
<td>1 000</td>
<td>68.10</td>
<td>0.25</td>
<td>0.17</td>
<td>-2.44</td>
<td>-5.53</td>
</tr>
<tr>
<td>5 000</td>
<td>78.56</td>
<td>-0.62</td>
<td>-0.63</td>
<td>-2.53</td>
<td>-5.44</td>
</tr>
<tr>
<td>10 000</td>
<td>82.29</td>
<td>-0.37</td>
<td>-0.36</td>
<td>-2.49</td>
<td>-4.44</td>
</tr>
</tbody>
</table>

Table 3: Average UAS difference for the subsets of the rich-resource treebanks under the no PoS tags setup.

<table>
<thead>
<tr>
<th># Sentences</th>
<th>$\text{rp}^h \times$</th>
<th>$\text{rx}^b \times$</th>
<th>$\text{2p}^b \times$</th>
<th>$\text{ah}^\text{tb} \times$</th>
<th>$\text{ct}^\text{tb} \times$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>35.60</td>
<td>0.06</td>
<td>9.31</td>
<td>7.57</td>
<td>4.83</td>
</tr>
<tr>
<td>500</td>
<td>58.63</td>
<td>3.04</td>
<td>2.45</td>
<td>0.99</td>
<td>-2.26</td>
</tr>
<tr>
<td>1 000</td>
<td>63.99</td>
<td>3.59</td>
<td>3.42</td>
<td>0.83</td>
<td>-2.24</td>
</tr>
<tr>
<td>5 000</td>
<td>75.57</td>
<td>1.47</td>
<td>1.55</td>
<td>-0.19</td>
<td>-3.07</td>
</tr>
<tr>
<td>10 000</td>
<td>79.90</td>
<td>1.22</td>
<td>1.54</td>
<td>-0.87</td>
<td>-2.93</td>
</tr>
</tbody>
</table>

Discussion The results from the experiments shed light about differences existing across different encodings and running configurations. First, under an ideal, gold environment, the $\text{rp}^h$ encoding makes a better use of limited data than the bracketing and transition-based encodings. Second, the predicted PoS tag setup shows that the performance of the PoS taggers can have a significant impact on the performance for the $\text{rp}^h$ encoding. More interestingly, weaknesses from different encodings seem to manifest to different extents depending on the amount of training data. For instance, when training data is scarce (100 sentences), bracketing encodings still cannot outperform the $\text{rp}^h$ encoding, despite the lower performance of the PoS taggers. However, when working with setups ranging from 500 to 1000 sentences, there is a slight advantage of the bracketing encodings with respect to $\text{rp}^h$, suggesting that with this amount of data, bracketing encodings could be the preferable choice, since they seem able to exploit their potential in a better way than the $\text{rp}^h$ encoding can exploit not fully accurate PoS tags. With more training samples, the relative PoS-based
encoding is again the best performing model across the board. In §4.2 we will discuss deeper how for truly low-resources languages the advantage in favour of bracketing representations exacerbates more for the predicted and no PoS tags setups.

<table>
<thead>
<tr>
<th></th>
<th>$r^b$</th>
<th>$r^h$</th>
<th>$2^b$</th>
<th>$ah^h$</th>
<th>$c^h$</th>
</tr>
</thead>
<tbody>
<tr>
<td>grc</td>
<td>83.09</td>
<td>79.89</td>
<td>81.7</td>
<td>78.57</td>
<td>79.86</td>
</tr>
<tr>
<td>lzh</td>
<td>90.21</td>
<td>89.56</td>
<td>89.24</td>
<td>89.04</td>
<td>89.18</td>
</tr>
<tr>
<td>cs</td>
<td>91.61</td>
<td>90.49</td>
<td>90.91</td>
<td>88.16</td>
<td>85.64</td>
</tr>
<tr>
<td>et</td>
<td>85.62</td>
<td>84.79</td>
<td>84.91</td>
<td>81.86</td>
<td>81.11</td>
</tr>
<tr>
<td>de</td>
<td>96.69</td>
<td>95.95</td>
<td>96.38</td>
<td>95.15</td>
<td>86.51</td>
</tr>
<tr>
<td>hi</td>
<td>94.69</td>
<td>94.09</td>
<td>94.43</td>
<td>93.05</td>
<td>85.02</td>
</tr>
<tr>
<td>ko</td>
<td>87.26</td>
<td>86.24</td>
<td>86.52</td>
<td>85.08</td>
<td>84.06</td>
</tr>
<tr>
<td>lv</td>
<td>85.3</td>
<td>83.88</td>
<td>84.01</td>
<td>80.88</td>
<td>81.38</td>
</tr>
<tr>
<td>fa</td>
<td>92.61</td>
<td>92.07</td>
<td>92.44</td>
<td>90.45</td>
<td>87.09</td>
</tr>
<tr>
<td>ro</td>
<td>90.49</td>
<td>89.68</td>
<td>89.03</td>
<td>87.39</td>
<td>86.38</td>
</tr>
<tr>
<td>ru</td>
<td>91.23</td>
<td>90.11</td>
<td>90.17</td>
<td>88.19</td>
<td>84.96</td>
</tr>
<tr>
<td>Avg</td>
<td>89.89</td>
<td>88.79</td>
<td>89.12</td>
<td>87.13</td>
<td>84.65</td>
</tr>
</tbody>
</table>

Table 4: UAS for the rich-resource treebanks, using the whole training set and the gold PoS tags setup. The red (-) and green cells (+++) show that a given encoding performed worse or better than the $r^b$ model, and that the difference is statistically significant. Lime and yellow cells mean that there is no a significant difference between a given encoding and the $r^b$, appending a + or a − when they performed better or worse than the $r^b$.

<table>
<thead>
<tr>
<th></th>
<th>$r^b$</th>
<th>$r^h$</th>
<th>$2^b$</th>
<th>$ah^h$</th>
<th>$c^h$</th>
</tr>
</thead>
<tbody>
<tr>
<td>grc</td>
<td>80.2</td>
<td>77.61</td>
<td>79.21</td>
<td>76.49</td>
<td>77.71</td>
</tr>
<tr>
<td>lzh</td>
<td>79.93</td>
<td>79.8</td>
<td>79.42</td>
<td>79.41</td>
<td>79.54</td>
</tr>
<tr>
<td>cs</td>
<td>90.04</td>
<td>88.93</td>
<td>89.34</td>
<td>86.67</td>
<td>84.25</td>
</tr>
<tr>
<td>et</td>
<td>81.07</td>
<td>80.36</td>
<td>80.34</td>
<td>77.71</td>
<td>76.95</td>
</tr>
<tr>
<td>de</td>
<td>95.85</td>
<td>95.14</td>
<td>95.34</td>
<td>94.34</td>
<td>85.79</td>
</tr>
<tr>
<td>hi</td>
<td>92.22</td>
<td>91.76</td>
<td>92.21</td>
<td>90.72</td>
<td>83.24</td>
</tr>
<tr>
<td>ko</td>
<td>84.25</td>
<td>83.44</td>
<td>83.42</td>
<td>82.98</td>
<td>81.25</td>
</tr>
<tr>
<td>lv</td>
<td>70.65</td>
<td>71.98</td>
<td>71.08</td>
<td>68.9</td>
<td>68.97</td>
</tr>
<tr>
<td>fa</td>
<td>90.30</td>
<td>89.8</td>
<td>90.32</td>
<td>88.89</td>
<td>85.28</td>
</tr>
<tr>
<td>ro</td>
<td>87.32</td>
<td>86.64</td>
<td>86.49</td>
<td>84.44</td>
<td>83.5</td>
</tr>
<tr>
<td>ru</td>
<td>88.71</td>
<td>88.13</td>
<td>88.24</td>
<td>85.93</td>
<td>82.96</td>
</tr>
<tr>
<td>Avg</td>
<td>85.51</td>
<td>84.87</td>
<td>85.06</td>
<td>83.26</td>
<td>80.86</td>
</tr>
</tbody>
</table>

Table 5: UAS for the rich-resource treebanks, using the whole training set and the predicted PoS tags setup.

Table 6: UAS for the rich-resource treebanks, using the whole training set and the no PoS tags setup.

<table>
<thead>
<tr>
<th></th>
<th>$r^b$</th>
<th>$r^h$</th>
<th>$2^b$</th>
<th>$ah^h$</th>
<th>$c^b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>af</td>
<td>88.02</td>
<td>85.7</td>
<td>85.48</td>
<td>81.84</td>
<td>78.6</td>
</tr>
<tr>
<td>cop</td>
<td>88.73</td>
<td>88.43</td>
<td>87.52</td>
<td>85.5</td>
<td>84.35</td>
</tr>
<tr>
<td>fo</td>
<td>84.04</td>
<td>83.76</td>
<td>84.09</td>
<td>81.78</td>
<td>79.53</td>
</tr>
<tr>
<td>hu</td>
<td>79.75</td>
<td>76.14</td>
<td>76.13</td>
<td>71.66</td>
<td>64.76</td>
</tr>
<tr>
<td>lt</td>
<td>51.98</td>
<td>50.28</td>
<td>50.19</td>
<td>45.0</td>
<td>46.6</td>
</tr>
<tr>
<td>mt</td>
<td>81.81</td>
<td>81.06</td>
<td>80.82</td>
<td>76.26</td>
<td>74.98</td>
</tr>
<tr>
<td>mr</td>
<td>77.43</td>
<td>76.46</td>
<td>75.97</td>
<td>76.94</td>
<td>73.54</td>
</tr>
<tr>
<td>ta</td>
<td>74.96</td>
<td>73.1</td>
<td>71.9</td>
<td>71.74</td>
<td>66.01</td>
</tr>
<tr>
<td>te</td>
<td>90.01</td>
<td>91.26</td>
<td>90.43</td>
<td>90.01</td>
<td>89.46</td>
</tr>
<tr>
<td>wo</td>
<td>86.19</td>
<td>84.04</td>
<td>84.51</td>
<td>80.65</td>
<td>77.43</td>
</tr>
<tr>
<td>Avg</td>
<td>80.29</td>
<td>79.08</td>
<td>78.82</td>
<td>76.19</td>
<td>73.48</td>
</tr>
</tbody>
</table>

Table 7: UAS for the low-resource treebanks for the gold PoS tags setup.

4.2 Experiment 2: Encodings performance on truly low-resource languages

**Data** We choose the 10 smallest treebanks (in terms of training sentences) that had a dev set: LithuanianHSE, MarathiUFAL, HungarianSzeged, TeluguMTG, TamilTTB, FaroeseFafPaHc, CopticScripturnium, MalteseMUDT, WolofWB and AfrikaansAfBooms (see Appendix B). Their sizes range between 153 and 1350 training sentences, most being around or between 500 and 1000 (see Appendix C).

**Setup** We rerun a subset of the experiments from §4.1, to check if the results follow the same trends, and conclusions are therefore similar.

**Results** Tables 7, 8 and 9 show the UAS for each encoding and treebank for the gold PoS tags setup, the predicted PoS tags setup and the no PoS tags setup, respectively. Again, under perfect conditions, the relative PoS-based encoding performs overall better, except for Telugu, which seems to

---

6Code switching treebanks and small treebanks of rich-resource languages were not considered.
be an outlier. For the predicted PoS tags setup, the bracketing-based encodings perform consistently better for most of the treebanks. For the no PoS tags setup, the bracketing-based encodings obtain, on average, more than 3 points than the relative PoS-head selection encoding, which even performs worse than the transition-based encodings.

**Discussion** These experiments help elaborate on the findings of §4.1. With respect to the ideal gold PoS tags setup, things do not change much, and the relative PoS-based encoding performs overall better. Still, this should not be taken as a ground truth about how the encodings will perform in real-world setups. For instance, for the predicted PoS tags setup, the bracketing-based encodings perform consistently better in most of the treebanks. This reinforces some of the suspicions found in the experiments of Table 2, where training on rich-resource languages, but with limited data, revealed that bracketing encodings performed better, although just slightly. Also, it is worth noting that most of the low-resource treebanks tested in this work have a number of training sentences in the range where the bracketing-based encodings performed better for the predicted PoS tags setup in Table 2, i.e. from 500 to 1000 sentences (see Appendix C). Yet, the better performance of bracketing encodings is more evident when running on real low-resource treebanks. This does not only suggest that the bracketing encodings are better for real low-resource sequence labeling parsing, but it could also pose more general limitations for other low-resource NLP tasks that are evaluated only on ‘faked’ low-resource setups, and that could lead to incomplete or even misleading conclusions.

Overall, the results suggest that bracketing encodings are the most suitable linearizations for real low-resource sequence labeling parsing.

**5 Conclusion**

We have studied sequence labeling encodings for dependency treebanks. First, we explored which encodings are more data-efficient under different conditions that include the use of gold PoS tags, predicted PoS tags and no PoS tags as part of the input. By restricting training data for rich-resource treebanks, we observe that although bracketing encodings are less data-efficient than head-selection ones under ideal conditions, this disadvantage can vanish when the input conditions are not gold and data is limited. Second, we studied their performance under the same running configurations, but on truly low-resource languages. These results show more clearly the greatest utility of bracketing encodings over the rest of the ones when training data is limited and the quality of external factors, such as PoS tags, is affected by the low-resource nature of the problem.

**Acknowledgements**

This work is supported by a 2020 Leonardo Grant for Researchers and Cultural Creators from the FBBVA.\(^{7}\) The work also receives funding from the European Research Council (FASTPARSE, grant agreement No 714150), from ERDF/MICINN-AEI (ANSWER-ASAP, TIN2017-85160-C2-1-R, SCANNER, PID2020-113230RB-C21), from Xunta de Galicia (ED431C 2020/11), and from Centro de Investigación de Galicia ‘CITIC’, funded by Xunta de Galicia and the European Union (European Regional Development Fund- Galicia 2014-2020 Program) by grant ED431G 2019/01.

---

\(^{7}\) FBBVA accepts no responsibility for the opinions, statements and contents included in the project and/or the results thereof, which are entirely the responsibility of the authors.

---

**Table 8:** UAS for the low-resource treebanks for the predicted PoS tags setup.

<table>
<thead>
<tr>
<th></th>
<th>$x^b$</th>
<th>$x^h$</th>
<th>$2p^b$</th>
<th>$ah^b$</th>
<th>$c^b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>af</td>
<td>81.84</td>
<td>80.29</td>
<td>79.0</td>
<td>77.3</td>
<td>73.61</td>
</tr>
<tr>
<td>cop</td>
<td>85.77</td>
<td>86.25</td>
<td>85.92</td>
<td>83.14</td>
<td>81.84</td>
</tr>
<tr>
<td>fo</td>
<td>77.04</td>
<td>76.97</td>
<td>77.52</td>
<td>75.23</td>
<td>74.24</td>
</tr>
<tr>
<td>hu</td>
<td>70.52</td>
<td>68.51</td>
<td>68.77</td>
<td>64.98</td>
<td>58.35</td>
</tr>
<tr>
<td>lt</td>
<td>30.28</td>
<td>34.53</td>
<td>33.11</td>
<td>31.23</td>
<td>29.91</td>
</tr>
<tr>
<td>mt</td>
<td>74.6</td>
<td>75.64</td>
<td>75.07</td>
<td>71.17</td>
<td>70.35</td>
</tr>
<tr>
<td>mr</td>
<td>66.99</td>
<td>67.96</td>
<td>67.23</td>
<td>68.93</td>
<td>67.23</td>
</tr>
<tr>
<td>ta</td>
<td>57.11</td>
<td>60.73</td>
<td>57.57</td>
<td>58.77</td>
<td>55.51</td>
</tr>
<tr>
<td>te</td>
<td>86.41</td>
<td>87.93</td>
<td>87.93</td>
<td>86.96</td>
<td>86.69</td>
</tr>
<tr>
<td>wo</td>
<td>76.88</td>
<td>76.4</td>
<td>76.3</td>
<td>73.24</td>
<td>70.84</td>
</tr>
<tr>
<td>Avg</td>
<td>70.74</td>
<td>71.52</td>
<td>70.93</td>
<td>69.10</td>
<td>66.86</td>
</tr>
</tbody>
</table>

**Table 9:** UAS for the low-resource treebanks for the no PoS tags setup.

<table>
<thead>
<tr>
<th></th>
<th>$x^b$</th>
<th>$x^h$</th>
<th>$2p^b$</th>
<th>$ah^b$</th>
<th>$c^b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>af</td>
<td>79.86</td>
<td>80.78</td>
<td>80.07</td>
<td>75.47</td>
<td>73.76</td>
</tr>
<tr>
<td>cop</td>
<td>84.36</td>
<td>85.76</td>
<td>85.13</td>
<td>83.07</td>
<td>81.28</td>
</tr>
<tr>
<td>fo</td>
<td>73.98</td>
<td>77.08</td>
<td>77.04</td>
<td>75.07</td>
<td>73.67</td>
</tr>
<tr>
<td>hu</td>
<td>63.63</td>
<td>65.21</td>
<td>64.87</td>
<td>62.04</td>
<td>56.17</td>
</tr>
<tr>
<td>lt</td>
<td>26.89</td>
<td>34.62</td>
<td>35.38</td>
<td>34.06</td>
<td>32.92</td>
</tr>
<tr>
<td>mt</td>
<td>70.95</td>
<td>75.5</td>
<td>75.3</td>
<td>71.69</td>
<td>70.32</td>
</tr>
<tr>
<td>mr</td>
<td>64.08</td>
<td>66.75</td>
<td>67.96</td>
<td>69.66</td>
<td>64.56</td>
</tr>
<tr>
<td>ta</td>
<td>52.79</td>
<td>60.03</td>
<td>56.61</td>
<td>59.58</td>
<td>54.95</td>
</tr>
<tr>
<td>te</td>
<td>85.44</td>
<td>88.49</td>
<td>88.63</td>
<td>87.1</td>
<td>86.82</td>
</tr>
<tr>
<td>wo</td>
<td>73.11</td>
<td>77.47</td>
<td>76.95</td>
<td>74.01</td>
<td>70.86</td>
</tr>
<tr>
<td>Avg</td>
<td>67.51</td>
<td>71.14</td>
<td>70.79</td>
<td>69.18</td>
<td>66.53</td>
</tr>
</tbody>
</table>
References


Kong, China. Association for Computational Linguistics.


### Appendix A  Taggers accuracy

<table>
<thead>
<tr>
<th># Sentences</th>
<th>Average accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>57.08</td>
</tr>
<tr>
<td>500</td>
<td>81.24</td>
</tr>
<tr>
<td>1000</td>
<td>85.03</td>
</tr>
<tr>
<td>5000</td>
<td>90.90</td>
</tr>
<tr>
<td>10000</td>
<td>92.77</td>
</tr>
<tr>
<td>Low-resource</td>
<td>85.03</td>
</tr>
</tbody>
</table>

Table 10: Average accuracy of the taggers for the splits of the rich-resource treebanks and the complete low-resource treebanks.

### Appendix B  Treebanks information

<table>
<thead>
<tr>
<th>% Non-projective sentences</th>
<th>Family</th>
<th>Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>de</td>
<td>6.76</td>
<td>IE (Germanic)</td>
</tr>
<tr>
<td>cs</td>
<td>11.49</td>
<td>IE (West Slavic)</td>
</tr>
<tr>
<td>ru</td>
<td>7.53</td>
<td>IE (East Slavic)</td>
</tr>
<tr>
<td>lzh</td>
<td>0.01</td>
<td>Sino-Tibetan</td>
</tr>
<tr>
<td>fa</td>
<td>14.22</td>
<td>IE (Iranian)</td>
</tr>
<tr>
<td>et</td>
<td>3.22</td>
<td>Uralic</td>
</tr>
<tr>
<td>ro</td>
<td>5.43</td>
<td>IE (Romance)</td>
</tr>
<tr>
<td>ko</td>
<td>21.70</td>
<td>Korean</td>
</tr>
<tr>
<td>grc</td>
<td>37.52</td>
<td>IE (Greek)</td>
</tr>
<tr>
<td>hi</td>
<td>13.60</td>
<td>IE (Indo-Aryan)</td>
</tr>
<tr>
<td>lv</td>
<td>6.53</td>
<td>IE (Baltic)</td>
</tr>
<tr>
<td>af</td>
<td>22.23</td>
<td>IE (Germanic)</td>
</tr>
<tr>
<td>cop</td>
<td>13.24</td>
<td>Afro-Asiatic</td>
</tr>
<tr>
<td>fo</td>
<td>0.19</td>
<td>IE (Germanic)</td>
</tr>
<tr>
<td>bu</td>
<td>27.10</td>
<td>Uralic</td>
</tr>
<tr>
<td>lt</td>
<td>14.07</td>
<td>IE (Baltic)</td>
</tr>
<tr>
<td>mt</td>
<td>3.86</td>
<td>Semitic</td>
</tr>
<tr>
<td>mr</td>
<td>6.01</td>
<td>IE (Indo-Aryan)</td>
</tr>
<tr>
<td>ta</td>
<td>1.67</td>
<td>Dravidian</td>
</tr>
<tr>
<td>te</td>
<td>0.15</td>
<td>Dravidian</td>
</tr>
<tr>
<td>wo</td>
<td>2.99</td>
<td>Niger-Congo</td>
</tr>
</tbody>
</table>

Table 11: Information about the treebanks used.

### Appendix C  Low-resource treebank sizes

<table>
<thead>
<tr>
<th># Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>AfrikaansAfriBooms</td>
</tr>
<tr>
<td>CopticScriptorium</td>
</tr>
<tr>
<td>FaroeseFarPaHC</td>
</tr>
<tr>
<td>HungarianTarg</td>
</tr>
<tr>
<td>LithuanianHSE</td>
</tr>
<tr>
<td>MalteseMUDT</td>
</tr>
<tr>
<td>MarathiUFAL</td>
</tr>
<tr>
<td>TamilTTB</td>
</tr>
<tr>
<td>TeluguMTG</td>
</tr>
<tr>
<td>WolofWTB</td>
</tr>
</tbody>
</table>

Table 12: Number of training sentences for the low-resource treebanks used.
Pre-training a BERT with Curriculum Learning by Increasing Block-Size of Input Text

Koichi Nagatsuka  
Soka University  
Tokyo, Japan  
e20d5301@soka-u.jp

Clifford Broni-Bediako  
Soka University  
Tokyo, Japan  
e18d5252@soka-u.jp

Masayasu Atsumi  
Soka University  
Tokyo, Japan  
matsumi@soka.ac.jp

Abstract

Recently, pre-trained language representation models such as BERT and RoBERTa have achieved significant results in a wide range of natural language processing (NLP) tasks; however, it requires extremely high computational cost. Curriculum learning (CL) is one of the potential solutions to alleviate this problem. CL is a training strategy where training samples are given to models in a meaningful order instead of random sampling. In this work, we propose a new CL method which gradually increases the block-size of input text for training the self-attention mechanism of BERT and its variants using the maximum available batch-size. Experiments in low-resource settings show that our approach outperforms the baseline in terms of convergence speed and final performance on down-stream tasks.

1 Introduction

Recent years have seen a series of breakthroughs in pre-trained language representation models. The development of pre-training methods like BERT (Devlin et al., 2019) and its variants (Liu et al., 2019) have led to large improvements in many down-stream tasks such as paraphrase identification, sentence textual similarity, sentiment analysis, and natural language inference. One of the advantages in training these models is that they can leverage the unlabeled large-scale corpora which are more available compared to the labeled ones. However, training these models with large-scale corpora is pretty expensive in terms of computational time and memory footprint. In the literature, there are three main approaches that have been adopted to address this problem. These are architecture-based approach (Sanh et al., 2019; Voita et al., 2019; Sukhbaatar et al., 2019; de Wynter and Perry, 2020; Lan et al., 2020), task-based approach (Yang et al., 2019; Clark et al., 2020) and dataset-based approach (Elman, 1993; Bengio et al., 2009; Moore and Lewis, 2010; Gururangan et al., 2020). While the architecture-based and task-based methods have been extensively studied in the context of pre-training methods for natural language processing (NLP), dataset-based approach is relatively unexplored. To this end, we adopt a dataset-based method called Curriculum Learning (CL) which controls the order of training samples so that the model might converge faster with better performance.

The idea of CL-like approach was originally proposed by Elman (1993). The idea is based on the actual learning mechanism of humans and animals, where basic concepts are acquired first, then more complex ones are gradually learned. Bengio et al. (2009) formalized this concept as CL to train neural networks. Through experimental analysis, Bengio et al. (2009) showed the benefit of CL on convergence speed and performance in shape recognition and language modeling tasks. One of the most significant challenges when adapting CL to a new task is to figure out a criterion for measuring the difficulty of the training samples. For example, in object recognition task, the size of objects is a good measure of difficulty (Shi and Ferrari, 2016; Ionescu et al., 2016), and the presence of low-frequent words in input text is an indicator of difficulty in language modeling (Bengio et al., 2009). These criteria vary greatly depending on the task, thus, it is not easy to define a measure of difficulty which is suitable for a particular task.

Most studies in the field of CL for NLP have proposed variety of difficulty measure by leveraging heuristics of the target tasks with neural networks (Bengio et al., 2009; Kocmi and Bojar, 2017; Soviany et al., 2021; Spitkovsky et al., 2009; Cirik et al., 2016; Rajeswar et al., 2017). On the other hand, it is not clear how to design CL for language representation models such as BERT. In pre-training BERT, distributed word representations are learned through optimizing masked language
modeling (MLM) loss which is computed by predicting a masked word or token in an input text. The input to the model is not a single sentence but an arbitrary-length span of text called a block. This indicates that it is not obvious how to measure the difficulty of the training samples using CL-based approach proposed in the previous studies.

The key component of BERT is the multi-head self-attention mechanism that learns to compute token embeddings from its context (Devlin et al., 2019). The multi-head self-attention mechanism can be thought as a problem of searching for important token-pairs based on the relative magnitude of attention among all the token-pairs in an input text. This process can be served as a clue which leads us to speculate that it might be possible to formulate CL strategy by focusing on the effective training of the self-attention mechanism in BERT. Although each individual head of the multi-head self-attention mechanism can learn any dependency among tokens, most of the heads tend to pay more attention to local dependencies than global ones (Kovaleva et al., 2019; Brunner et al., 2019; Sukhbaatar et al., 2019; Jiang et al., 2020). It could be easier to train local self-attention in shorter blocks of input text than global self-attention in longer ones. Therefore, the block-size of input text can be used as the effective criterion to measure the difficulty-level of training samples for BERT.

In this paper, we introduce a new CL method which gradually increases the block-size of input text for pre-training BERT using the maximum available batch-size to accomplish convergence speedup, and also improve performance in the down-stream tasks. Since our approach is very simple, it is easy to apply it to BERT and its variants with little effort. Using a small-scale corpus, the experimental results demonstrated that our proposed approach outperforms the baseline on GLUE tasks with faster convergence speed.

2 Related Work

To reduce the memory footprint and improve the training speed of pre-trained language models, prior works have shown that architecture-based approaches are very useful. Sanh et al. (2019) proposed to leverage knowledge distillation to train a smaller version of BERT with faster training speed while maintaining comparative performance. Lan et al. (2020) used factorized embedding parameterization and cross-layer parameter sharing, which led to the reduction of parameter size and training time. de Wynter and Perry (2020) applied neural architecture search to select the optimal architecture of BERT and successfully compressed the size of the model. Task-based approaches have also been explored for pre-training language models with high training efficiency. Yang et al. (2019) introduced permutation language modeling which retains the benefits of autoregressive models and allows the models to capture bidirectional context. Instead of performing pre-training with MLM task, Clark et al. (2020) trained a BERT as a discriminator that determines whether each corrupted token was replaced by a generator model.

Recent studies have shown that CL is a successful approach for a wide range of machine learning applications (Soviany et al., 2021; Wang et al., 2021), including the fine-tuning of large-scale language models such as BERT (Xu et al., 2020). Some of large-scale language models like GPT-3 (Brown et al., 2020) and T5 (Raffel et al., 2020) adopted non-uniform mixing strategies which control the amount of training samples from multiple corpora. However, CL strategy has not directly been applied to pre-training large-scale language models. There exists many studies of CL which used the length of sentences or input sequences as a measure of difficulty in NLP tasks including neural machine translation (Kocmi and Bojar, 2017), sentiment analysis (Cirik et al., 2016), parsing (Spitkovsky et al., 2009), poem generation (Rajeswar et al., 2017) and reading comprehension task (Tay et al., 2019). In this work, we exploit the block-size of input text in the context of self-attention mechanism as a measure of difficulty for pre-training BERT.

3 Method

The overview of the proposed CL method is presented in Figure 1. The method is divided into two stages: (a) Splitting a corpus based on specific block-sizes and (b) Gradual training of BERT by increasing the block-size. In the first stage, we split the original corpus into a series of input blocks with the pre-defined length. In the second stage, we train a model by changing the training samples from the short block-size to the long one depending on the pre-defined number of training steps. In training, some tokens in a block are randomly masked to perform the MLM task. We describe the MLM task and the details of the two stages of our
CL approach in this section.

3.1 Masked Language Modeling (MLM)

Let \( x = x_1, x_2, \ldots, x_T \) denotes a sequence of original tokens, where \( T \) is a block-size. By randomly masking an arbitrary number of tokens, we obtain an input sequence \( \hat{x} \). Given the corrupted sequence \( \hat{x} \), MLM is a task of predicting the original sequence \( x \). The training objective is formulated as:

\[
\max_{\theta} \log p_\theta(x | \hat{x}) \approx \sum_{i=1}^{T} m_i \log p_\theta(x_i | x_{<i}, x_{>i})
\]

where \( x_i \) is the predicted token at position \( i \) and \( \theta \) is the parameters of a model. \( m_i \) indicates the presence of a masked token where \( m_i = 1 \) if \( x_i \) is masked, otherwise \( 0 \). For this objective, we optimize the model parameters using the cross-entropy loss. In the MLM task, models infer masked tokens from bi-directional context \( (x_{<i} \text{ and } x_{>i}) \). The block-size restricts the available context information in both directions and thus affects the MLM accuracy.

3.2 Splitting a Corpus Based on Block-sizes

In the first stage, we split the original corpus into training samples with the specified size. Each input text for training BERT is not a linguistically coherent unit like a sentence or multiple sentences, but a fixed span of contiguous text (Devlin et al., 2019) that we called a block. In other words, the input is not guaranteed to end with a period nor start with a first word in a sentence. Liu et al. (2019) argues that it is desirable to acquire the input sequence to be at most 512 tokens through the extensive experiments. We follow this setting to obtain the block of a specified length from the corpus as a training sample. We train a byte-level Byte-Pair Encoding (BPE) tokenizer as in (Radford et al., 2019) to split the raw text into a sequence of tokens. By using byte-level BPE, we can decompose all words including out-of-vocabularies, which are likely to appear at test time especially when using a small training dataset. In the experiment, we set the vocabulary size to 20,000.

3.3 Gradual Training

In the second stage, we train a model step-by-step with four different block-sizes which are 64, 128, 256, and 512. We first train the model with the shortest block-size, which is 64 in this case, for an arbitrary number of steps. Then, we retrain the model in the order of 128 and 256 block-sizes respectively for the same number of steps. Finally, we retrain the model with the longest block-size of 512 until it converges. For masking tokens, we use the fixed masking rate of 0.15. When restarting the training, we always initialize the learning rate. To accelerate training, we use the maximum available batch-size depending on the block-size. Since our proposed method is designed to limit the block-size in the early training phase, we employ larger batch-size with shorter block-size which improves the whole training efficiency.

4 Experiments

In the experiments, we evaluate our proposed CL approach in terms of the convergence speed and model performance. We use wikinet-2 (Merity et al., 2019)
Figure 2: Comparison of our approach and the baseline on the validation losses. Left (a): The result of CL which increases the block-size with the maximum available batch-size. Right (b): The result of CL which increases block-sizes with the fixed batch size (16). Black dotted lines indicates the points where the block-size of training samples is changed, and the red dotted line indicates each convergence point.

<table>
<thead>
<tr>
<th>Model (block-size)</th>
<th>Training time</th>
<th>Number of steps</th>
<th>Memory (batch-size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline(512)</td>
<td>5:28:38</td>
<td>60K</td>
<td>17.5(16)</td>
</tr>
<tr>
<td>Curriculum(64)</td>
<td>1:19:15</td>
<td>10K(fixed)</td>
<td>19.2(256)</td>
</tr>
<tr>
<td>Curriculum(128)</td>
<td>1:21:02</td>
<td>10K(fixed)</td>
<td>21.1(128)</td>
</tr>
<tr>
<td>Curriculum(256)</td>
<td>1:07:16</td>
<td>10K(fixed)</td>
<td>19.9(48)</td>
</tr>
<tr>
<td>Curriculum(512)</td>
<td>0:50:10</td>
<td>10K</td>
<td>17.5(16)</td>
</tr>
<tr>
<td><strong>Curriculum(total)</strong></td>
<td><strong>4:37:43</strong></td>
<td><strong>40K</strong></td>
<td>—</td>
</tr>
</tbody>
</table>

Table 1: Statistics on training of the baseline and each curriculum training phase.

et al., 2016) for pre-training RoBERTa (Liu et al., 2019), which is a variant of BERT. For fine-tuning on down-stream tasks, we use the General Language Understanding Evaluation (GLUE) dataset (Wang et al., 2018). All the training and fine-tuning were carried out on a GeForce RTX3090 with 24GB memory.

4.1 Datasets

Wikitext-2: Although BERT and its variants (e.g. RoBERTa) are commonly trained with large-scale corpora which contain over 3 billion words, we use wikitext-2 (Merity et al., 2016) which is a small corpus to enable pre-training with a limited computational resource. Wikitext-2 is one of the standard corpora for language models, and consists of 720 good-quality articles from English Wikipedia. It has about 2M tokens for training, and 217K and 245K tokens for validation and testing respectively.

GLUE Benchmarks: We fine-tune our models on the GLUE benchmarks (Wang et al., 2018). GLUE consists of nine datasets for measuring the generalization performance of pre-trained language models. We use only 7 datasets (SST-2, MRPC, QQP, MNLI-m, QNLI, RTE, and WNLI) out of the 9 GLUE benchmarks. CoLA and STS-B are removed due to a tendency to fall into over-fitting which stems from the small-scale pre-training.

4.2 Training Details

We perform both curriculum training and anti-curriculum training in the pre-training of RoBERTa. In curriculum training, we increase the block-size of training samples from the shortest to the longest. On the other hand, in anti-curriculum training, training samples with the longest block-size are first given to the model as the most difficult ones, then the difficulty-level of training samples is gradually reduced by shortening the block-size in the training process. By comparing curriculum training with anti-curriculum training, which follows the opposite sampling order, we show that increasing block-size is an effective CL method for pre-trained language representation models.

For all the models, we use the same RoBERTa-base architecture which has 12 layers with a hidden size of 768. Each layer has 12 attention heads. We
use AdamW (Loshchilov and Hutter, 2017) with a learning rate of 1e-5 in the pre-training with four different batch-sizes depending on the block-sizes as shown in Table 2. In fine-tuning, we also use the same optimizer as used in pre-training and set a learning rate to 5e-5 and batch-size to 64 for all task except for QNLI where we use learning rate of 2e-5 and batch-size of 16 due to the memory limitation.

We define the training time of the overall curriculum training as a total of the training time for every training phase corresponding to each block-size. In both curriculum training and anti-curriculum training, our models are trained for 10,000 steps with each block-size except for the last block-size where we continue the training until the convergence of the models. For comparative evaluation, we train RoBERTa without CL by using random sampling as the baseline model.

### 4.3 Results

#### 4.3.1 Convergence Speed

Figure 2(a) shows the comparison of our curriculum model which increases the block-size with the maximum available batch-size and the baseline on the validation losses throughout pre-training. Compared to the loss of the baseline model that converged at around 5.0, the loss of curriculum model decreased steadily and achieved the faster convergence speed outperforming the baseline by about 2 points in validation loss. The learning curve of the baseline model were plateau until 35K steps, and then the loss finally restarted to descend. On the other hand, the loss of the curriculum model stably decreased every time we switched the difficulty-level of training samples. To analyze the effect of increasing a batch-size on convergence speed, we demonstrated an ablation study by fixing the batch-size to 16 (which is the maximum size when block-size is set to 512). Figure 2(b) shows the result of the curriculum model which increases block-size with the fixed batch-size. Compared to the our proposed curriculum model (Figure 2(a)), it required about 60K steps to converge, which is the same training time as the baseline. This result indicates that CL improves final performance but does not contribute to the convergence speedup in case the batch-size is fixed.

Table 1 presents the statistical information about the training of the baseline and each curriculum phase. While the baseline model converged after about 60K steps, our curriculum model required just 40K steps in total, which is about 1.5 times faster than the baseline. Although using the large batch-size depending on the small block-size tended to take long training time, it allows training a large number of training samples and the total training time was reduced by about 1.0 hours. Table 2 represents the comparison of training efficiency between the baseline and our curriculum model. With respect to the training samples per second, curriculum model achieved better training efficiency, which is 5 times as higher as the baseline, and also resulted in much better validation loss.

#### 4.3.2 GLUE Results

Table 3 shows the GLUE scores on development datasets. For all 6 down-stream tasks, our curriculum model at the bottom of the table outperformed the baseline model at the top. Especially, performances on STS-2, MRPC, QQP, MNLI-m and QNLI were higher than the baseline by a large margin (+4.47 on SST-2, +3.19 on MRPC, +6.48 F1 score and +3.37 accuracy on QQP, 8.89 on MNLI-m, and 15.74 on QNLI) while accuracy on RTE and WNLI were extremely low in both curriculum and baseline. Although each scores of our model is not high due to the small-scale pre-training, relative improvements of scores by CL were generally observed.
<table>
<thead>
<tr>
<th>Model</th>
<th>(The Order of block-size)</th>
<th>SST-2</th>
<th>MRPC</th>
<th>QQP (F1/Acc.)</th>
<th>MNLI-m</th>
<th>QNLI</th>
<th>RTE</th>
<th>WNLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>(512)</td>
<td>79.01</td>
<td>69.60</td>
<td>69.77</td>
<td>57.39</td>
<td>63.97</td>
<td>51.98</td>
<td>56.33</td>
</tr>
<tr>
<td>Anti-Curriculum</td>
<td>(512, 256, 128, 64)</td>
<td>80.38</td>
<td>70.09</td>
<td>72.88</td>
<td>60.64</td>
<td>49.46</td>
<td>52.34</td>
<td>56.33</td>
</tr>
<tr>
<td>Curriculum w/o 64</td>
<td>(128, 256, 512)</td>
<td>81.99</td>
<td>69.60</td>
<td>74.81</td>
<td>64.97</td>
<td>78.74</td>
<td>47.29</td>
<td>46.47</td>
</tr>
<tr>
<td>Curriculum w/o 128</td>
<td>(64, 256, 512)</td>
<td>83.37</td>
<td>70.58</td>
<td>75.21</td>
<td>75.21</td>
<td>77.74</td>
<td>45.12</td>
<td>56.33</td>
</tr>
<tr>
<td>Curriculum w/o 256</td>
<td>(64, 128, 512)</td>
<td>82.45</td>
<td>70.34</td>
<td>75.76</td>
<td>65.76</td>
<td>77.75</td>
<td>50.18</td>
<td>46.47</td>
</tr>
<tr>
<td>Curriculum w/o 512</td>
<td>(64, 128, 256)</td>
<td>80.61</td>
<td>70.83</td>
<td>75.76</td>
<td>66.53</td>
<td>75.76</td>
<td>51.26</td>
<td>32.39</td>
</tr>
<tr>
<td>Curriculum 2-stage</td>
<td>(64, 512)</td>
<td>80.84</td>
<td>72.05</td>
<td>76.21</td>
<td>66.82</td>
<td>77.22</td>
<td>48.73</td>
<td>56.33</td>
</tr>
<tr>
<td>Curriculum (Ours)</td>
<td>(64, 128, 256, 512)</td>
<td><strong>83.48</strong></td>
<td><strong>72.79</strong></td>
<td><strong>76.25</strong></td>
<td>66.28</td>
<td><strong>79.71</strong></td>
<td><strong>53.42</strong></td>
<td>56.33</td>
</tr>
</tbody>
</table>

Table 3: GLUE scores on development datasets. batch-size=64, lr=5e-5, but in QNLI, batch-size=16, lr=2e-5.

4.3.3 Comparison with Anti-Curriculum

Compared with our curriculum model, performances of anti-curriculum model were lower on every down-stream tasks. This result indicates that not decreasing but increasing a block-size is the effective for improving the generalization performances. Interestingly, the performances of anti-curriculum were better or equal to the baseline in all tasks except for QNLI. One possible reason for this result is that generating training samples with various block-sizes may have the same impact as data augmentation. Anti-curriculum model, however, failed to learn the QNLI task because the model is optimized for short text like 64 tokens at the end while the input of QNLI contains samples whose input length is longer than 64.

4.3.4 Ablation Study

As an ablation study, we tested two types of models including 3-stage curriculum and 2-stage curriculum. For the 3-stage curriculum, we removed a specific block-size from our training schedule and conducted the CL with the rest of block-sizes. For 2-stage curriculum, we trained the model only with the shortest block-size (64 tokens) and longest one (512 tokens).

As Table 3 shows, our curriculum model with the full training schedule is equal to or slightly better than the 2-stage or 3-stage models on each down-stream tasks. However, for tasks where performance gaps are not significant, the 2-stage and 3-stage curricula are more advantageous because of the shorter training time. As in the case of the anti-curriculum, the curriculum model without block-size of 512 tokens, that was not optimized for the largest block-size, had lower performance in QNLI. The 2-stage curriculum, which requires the least amount of training time, achieved almost the same accuracy as the normal curriculum in MRPC and MNLI-m, but relatively poor performance in tasks such as SST-2. These experiments show that there is room to further speed-up of CL by modifying the curriculum schedule on the block-size. Moreover, the result also indicates that the impact of CL on the performance will be different depending on each down-stream task.

5 Conclusion

In this paper, we proposed a new CL method for pre-training BERT, which progressively increase a block-size of input text. Our approach is very simple and thus handy to implement. Experiments in the low-resource setting have shown that proposed method leads to faster convergence speed and better performances in down-stream tasks. In further research, we expand the corpus and validate the scalability of our approach. In addition, we speculate that it is important to investigate when the difficulty-level should be changed through the training and how it affect model performances.
References


COVID-19 in Bulgarian Social Media: Factuality, Harmfulness, Propaganda, and Framing

Preslav Nakov,¹ Firoj Alam,¹ Shaden Shaar,¹ Giovanni Da San Martino² and Yifan Zhang¹
¹ Qatar Computing Research Institute, HBKU, Qatar
² University of Padova, Italy

{pnakov, fialam, sshaar, yzhang}@hbku.edu.qa, dasan@math.unipd.it

Abstract

With the emergence of the COVID-19 pandemic, the political and the medical aspects of disinformation merged as the problem got elevated to a whole new level to become the first global infodemic. Fighting this infodemic is currently ranked very high on the list of priorities of the World Health Organization, with dangers ranging from promoting fake cures, rumors, and conspiracy theories to spreading xenophobia and panic. With this in mind, we studied how COVID-19 is discussed in Bulgarian social media in terms of factuality, harmfulness, propaganda, and framing. We found that most Bulgarian tweets contain verifiable factual claims, are factually true, are of potential public interest, are not harmful, and are too trivial to fact-check; moreover, zooming into harmful tweets, we found that they spread not only rumors but also panic. We further analyzed articles shared in Bulgarian partisan pro/con-COVID-19 Facebook groups and found that propaganda is more prevalent in skeptical articles, which use doubt, flag waving, and slogans to convey their message; in contrast, concerned ones appeal to emotions, fear, and authority; moreover, skeptical articles frame the issue as one of quality of life, policy, legality, economy, and politics, while concerned articles focus on health & safety.

1 Introduction

The ongoing global COVID-19 pandemic has brought an unprecedented situation with a lot of uncertainty: as this was a new disease, very little was known about it. This created an information void, where there was a lot of demand but little supply of reliable new information: a perfect breeding ground for all kinds of rumors and conspiracy theories, whose spread was facilitated by social media, which in turn optimized for user engagement (yet, later, they did put serious efforts in trying to limit the spread of false claims about COVID-19).

Unlike previous events that attracted a lot of disinformation, the emergence of the COVID-19 pandemic gave rise to a new powerful blending of medical and political disinformation, which resulted in the first global infodemic. Indeed, shortly after having declared the COVID-19 outbreak a pandemic, the World Health Organization had to engage in counter-measures against the growing infodemic, which it ranked among its top priorities in the fight against the COVID-19 pandemic.¹

Figure 1 shows some tweets that demonstrate how COVID-19 is discussed in Bulgarian social media. We can see that the problem goes beyond factuality: while some tweets spread rumors (Figure 1a), other discuss cure (Figure 1b). Indeed, the infodemic quickly extended to promoting bad cure, instilling panic, xenophobia, racism, and distrust in authorities, among others. (Alam et al., 2021b)

1https://www.who.int/health-topics/infodemic

Figure 1: Bulgarian tweets with English translation.

(a) rumor

I just saw Край на споровете за произхода на COVID-19! Той е изкуствено създаван от COVID-19 през 2015 г. - Click to see also ⬇
thebulgariantimes/%d0%9a%d1%80%d...

(b) discusses cure

Открыто е първото лекарство, което доказано спасява от COVID-19 https://t.co/rUx9EA05gx

The first drug that has been proven to save from COVID-19 has been found
https://t.co/rUx9EA05gx

Figure 1: Bulgarian tweets with English translation.
Thus, it is important to analyze social media posts in terms of factuality, harmfulness, checkworthiness, etc. It is also useful to understand whether the post is propagandistic, what propaganda techniques are used, and how the issue is framed. While there have been studies focusing on (some of) these issues for high-resource languages such as English and Arabic (Barrón-Cedeño et al., 2020; Hossain et al., 2020; Li et al., 2020; Alam et al., 2021b; Nakov et al., 2021a,c), there has been less work for low-resource languages such as Bulgarian (Dinkov et al., 2019; Alam et al., 2021d; Shaar et al., 2021b,c). Here, we aim to bridge this gap by analyzing tweets and Facebook posts about COVID-19 in Bulgarian, with focus on factuality, harmfulness, propaganda, and framing.

Our contributions can be summarized as follows:

- We create a dataset of tweets and Facebook posts related to COVID-19.\(^2\)
- We perform analysis from various perspectives (factuality, harmfulness, propaganda, and framing), and we discuss some interesting observations from our analysis.

The rest of the paper is organized as follows: Section 2 offers a brief overview of previous work. Section 3 describes the dataset. Section 4 discusses our methodology. Section 5 discusses the findings. Finally, Section 7 concludes and points to possible directions for future work.

2 Related Work

Below, we discuss work relevant to our analysis, focusing on factuality, check-worthiness, propaganda, framing, and fighting the COVID-19 infodemic.

2.1 Factuality

A variety of task formulations have been proposed to address the spread of misinformation and disinformation online, and for each formulation, a number of approaches have been developed. Some good readings on the topic include surveys such as that by Shu et al. (2017), who adopted a data mining perspective on “fake news” and focused on social media. Another survey (Zubiaga et al., 2018) studied rumor detection in social media. The survey by Thorne and Vlachos (2018) took a fact-checking perspective on “fake news” and related problems. Li et al. (2016) covered truth discovery in general. Lazer et al. (2018) offered an overview and discussion on the science of “fake news”. Vosoughi et al. (2018) focused on the proliferation of true and false news online. Other recent surveys focused on stance detection (Küçük and Can, 2020), propaganda (Nakov et al., 2021b), social bots (Ferrara et al., 2016), false information (Zannettou et al., 2019), and bias on the Web (Baesa-Yates, 2018).

Some very recent surveys featured stance for misinformation and disinformation detection (Hardalov et al., 2021), automatic fact-checking to assist human fact-checkers (Nakov et al., 2021b), predicting the factuality and the bias of entire news outlets (Nakov et al., 2021d), and multimodal disinformation detection (Alam et al., 2021a). A large body of research has focused on developing automatic systems for fact-checking to limit the spread of disinformation and misinformation (Li et al., 2016; Hardalov et al., 2016; Shu et al., 2017; Lazer et al., 2018; Mihaylova et al., 2018; Vosoughi et al., 2018; Nguyen et al., 2020). This includes development of datasets (Wang, 2017; Augenstein et al., 2019), and organizing evaluation campaigns (Derczynski et al., 2017; Nakov et al., 2018; Da San Martino et al., 2019; Elsayed et al., 2019; Gorrell et al., 2019; Mihaylova et al., 2019; Barrón-Cedeño et al., 2020; Nakov et al., 2021c; Shaar et al., 2021b). However, there are credibility issues with automated systems (Arnold, 2020). Hence, another research direction has emerged: building tools to facilitate human fact-checkers (Nakov et al., 2021b).

2.2 Check-Worthiness Estimation

Most work on check-worthiness focused on political debates and speeches. This includes the ClaimBuster (Hassan et al., 2015) and the ClaimRank systems (Jaradat et al., 2018), shared tasks at CLEF (Atanasova et al., 2018, 2019, 2020, 2021c), modeling the context of the claim (Gencheva et al., 2017; Patwari et al., 2017; Shaar et al., 2021a), and multi-task learning from the decisions of multiple fact-checking organizations (Vasileva et al., 2019).

There has been less research on identifying check-worthy claims in social media posts. Previous work in this direction includes check-worthiness estimation of COVID-19 and political tweets (Alam et al., 2021d,b; Shaar et al., 2020, 2021b,c).
More directly related to our work here is the work of Alam et al. (2021d) and Alam et al. (2021b), who developed a multi-question annotation schema to annotate tweets about COVID-19, organized around seven questions that model the perspective of journalists, fact-checkers, social media platforms, policymakers, and the society. In our experiments, we use their schema and data to train classifiers for part of our analysis.

2.3 Propaganda

Propaganda is a communication tool, deliberately designed to influence the opinions and the actions of other people in order to achieve a predetermined goal. Computational propaganda is defined as the use of automated approaches to intentionally disseminate misleading information on social media platforms (Woolley and Howard, 2018).

Most research on propaganda detection has focused on analyzing textual content (Barrón-Cedeno et al., 2019; Rashkin et al., 2017; Da San Martino et al., 2019, 2020a). Rashkin et al. (2017) developed the TSHP-17 corpus, which uses document-level annotation and is labeled with four classes: trusted, satire, hoax, and propaganda. They trained a model using word n-gram representation with logistic regression and reported that the model performed well only on articles from sources that the system was trained on. Barrón-Cedeno et al. (2019) developed the QProp corpus with two labels: propaganda vs. non-propaganda. They also experimented on TSHP-17 and QProp corpora, where for the TSHP-17 corpus, they binarized the labels: propaganda vs. any of the other three categories. Similarly, Habernal et al. (2017, 2018) developed a corpus with 1.3k arguments annotated with five fallacies, including ad hominem, red herring, and irrelevant authority, which directly relate to propaganda techniques.

A more fine-grained propaganda analysis was done by Da San Martino et al. (2019), who developed a corpus of news articles annotated with 18 propaganda techniques. Subsequently, the Prta system was released (Da San Martino et al., 2020b), and improved models were proposed, focusing on interpretability (Yu et al., 2021) or addressing the limitations of transformers (Chernyavskiy et al., 2021). Very recently, multimodal content was explored in memes using 22 fine-grained propaganda techniques (Dimitrov et al., 2021a,b).

2.4 Framing

Framing is a strategic device and a central concept in political communication, for representing different salient aspects and perspectives for the purpose of conveying the latent meaning about an issue (Entman, 1993). It is important for news media as the same topics can be discussed from different perspectives, which can influence our understanding, beliefs, and attitudes regarding what is happening in our society. There has been recent work on automatically identifying media frames, which includes developing coding schemes and datasets such as the Media Frames Corpus (Card et al., 2015), developing systems to automatically detect media frames (Liu et al., 2019; Zhang et al., 2019), large scale automatic analysis of New York Times Articles (Kwak et al., 2020), and a semi-supervised approach to detecting frames in online news sources (Cheeks et al., 2020).

2.5 COVID-19 Research

Since the beginning of the COVID-19 pandemic, there has been a large number of work on fighting the COVID-19 infodemic. Most notable work includes developing multi-question annotation schemas of tweets about COVID-19 (Alam et al., 2021d,b), studying credibility (Cinelli et al., 2020; Pulido et al., 2020; Zhou et al., 2020), racial prejudices and fear (Medford et al., 2020; Vidgen et al., 2020), situational information, e.g., caution and advice (Li et al., 2020), as well as on detecting mentions and stance with respect to known misconceptions (Hossain et al., 2020).

Another less relevant research line is on the development of datasets of tweets about COVID-19 (Cinelli et al., 2020; Song et al., 2021; Zhou et al., 2020; Haouari et al., 2021)

3 Dataset

Tweets: Using the Twitter API, we collected 30k tweets from January 2020 till November 2020. We performed search by specifying the target language to be Bulgarian and asking for the tweet to contain the following keywords and hashtags related to COVID-19 (English translations are shown in bleu color):

#корона, #коронавирус, коронавирус, корона
#corona, #coronavirus, coronavirus, corona
We only selected original tweets (no retweets or replies), we removed duplicates using a similarity-based approach (Alam et al., 2021c), and we filtered out tweets with less than five words. Finally, we selected the most frequently liked and retweeted tweets for annotation. For our analysis, we manually annotated 4k of them using the multi-question annotation schema from (Alam et al., 2021b), with three annotators per tweet (a total of 11k annotations). This Bulgarian data is also used in (Alam et al., 2021d) and for the CLEF 2021 CheckThat! lab task 1 (Shaar et al., 2021).

**Articles in Facebook posts:** We further collected articles posted in Bulgarian Facebook groups that discuss COVID-19. We focused on concerned, skeptical, and conspiracy groups; the list is shown in Figure 3. We collected the links to articles posted in these groups, and we manually annotated each article as skeptical or concerned.

**4 Method**

Figure 2 shows our analysis pipeline. Below, we discuss each element of the pipeline in more detail.

**4.1 Manual Annotation**

The manual tasks consist of multi-question disinformation annotation of tweets and also of skeptical vs. concerned articles posted on Facebook.

**Concerned**

(10.9k) Covid-19: факти срещу слухове
(10.9k) Covid-19: facts against rumors
(2.4k) Коронавирус/COVID-19 - само валидирана информация
(2.4k) Coronavirus / COVID-19 - validated information only
(1.0k) Здрав разум за здрава държава
(1.0k) Common sense for a healthy state

**Skeptical**

(14.8k) Аз подкрепям доцент Мангъров
(14.8k) I support Associate Professor Mangarov
(6.5k) Подкрепя за Атанас Мангъров
(6.5k) Support for Atanas Mangarov
(4.0k) С доц. д-р Манг*р*в и Бог обратно в живота
(4.0k) With Assoc. Prof. Dr. Mang * r * v and God back to life

**Conspiracy**

(1.4k) ИСТИНATA : Коронавирус / COVID-19, КОВИД-19/
(1.4k) THE TRUTH: Coronavirus / COVID-19, COVID-19/
(1.3k) Измислицата Корона- вирус
(1.3k) The Corona virus
(1.2k) Измамата Ковид19
(1.2k) The Kovid Deception19
(0.2k) Корона вирус COVID-19 или големата манипулация
(0.2k) Crown virus COVID-19 or major manipulation

Figure 3: Facebook groups we collected articles from.

**4.1.1 Disinformation Annotation for Tweets**

For the disinformation analysis, we used the holistic approach in (Alam et al., 2021b). It is formulated into seven questions, asking whether a tweet (Q1) contains a verifiable factual claim, (Q2) is likely to contain false information, (Q3) is of interest to the general public, (Q4) is potentially harmful to a person, a company, a product, or society, (Q5) requires verification by a fact-checker, (Q6) poses harm to society and why, or (Q7) requires the attention of policy makers and why. Three annotators worked on each tweet, following the annotation guidelines in (Alam et al., 2021b).
The annotators were fluent in Bulgarian, two were male and one was female, with qualifications ranging from undergraduates to people with a MSc degree. For disagreed annotations, a final consolidator participated in the discussion to decide the final label. We computed the inter-annotator agreement between the annotators and the final consolidated label using Fleiss Kappa ($\kappa$) as shown in Table 1. We can see that there was moderate to substantial agreement between the human annotators across the questions, according to the range of values for $\kappa$ suggested in (Landis and Koch, 1977).

<table>
<thead>
<tr>
<th>Agree. Pair</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 - C</td>
<td>0.77</td>
<td>0.44</td>
<td>0.64</td>
<td>0.53</td>
<td>0.49</td>
<td>0.53</td>
<td>0.51</td>
</tr>
<tr>
<td>A2 - C</td>
<td>0.51</td>
<td>0.40</td>
<td>0.59</td>
<td>0.49</td>
<td>0.44</td>
<td>0.56</td>
<td>0.53</td>
</tr>
<tr>
<td>A3 - C</td>
<td>0.47</td>
<td>0.38</td>
<td>0.57</td>
<td>0.49</td>
<td>0.38</td>
<td>0.53</td>
<td>0.40</td>
</tr>
<tr>
<td>Avg</td>
<td>0.58</td>
<td>0.41</td>
<td>0.60</td>
<td>0.50</td>
<td>0.44</td>
<td>0.54</td>
<td>0.48</td>
</tr>
</tbody>
</table>

Table 1: Inter-annotator agreement using Fleiss Kappa for the 7-level annotation for disinformation in tweets.

4.1.2 Skeptical vs. Concerned Annotation for Articles Posted on Facebook

The same annotators further annotated the Facebook articles as skeptical or concerned. This was a fairly straightforward task, with almost no disagreement. Note that we analyzed each article manually to decide whether it is skeptical or concerned (rather than using distant supervision to propagate the label for the group to label articles automatically, even though the vast majority of articles could be labeled with the label of the group).

4.2 Automatic Classification

For the analysis of propaganda and framing, both for tweets and for news articles, we used the automatic models discussed below.

4.2.1 Propaganda Analysis

For this analysis, we used Proppy and Prta.

Proppy (Barrón-Cedeño et al., 2019) uses a fragment-level and a sentence-level classifier. They were trained on a corpus of 350K tokens. The performance of the sentence-level classifier is 60.71 in terms of F1 score. The fragment-level classifier identifies the text fragments and the propaganda techniques that occur in them. They consider the following 18 techniques: (i) loaded language, (ii) name calling or labeling, (iii) repetition, (iv) exaggeration or minimization, (v) doubt, (vi) appeal to fear/prejudice, (vii) flagwaving, (viii) causal oversimplification, (ix) slogans, (x) appeal to authority, (xi) black-and-white fallacy, dictatorship, (xii) thought-terminating cliché, (xiii) whataboutism, (xiv) reductio ad Hitlerum, (xv) red herring, (xvi) bandwagon, (xvii) obfuscation, intentional vagueness, confusion, and (xviii) straw man.

Note that both Proppy and Prta only support English. To prepare their input, we translated the Bulgarian text to English using Google.

4.2.2 Framing

We used the Tanbih Framing Bias Detection system (Zhang et al., 2019), trained on the Media Frames Corpus (11k training news articles) by fine-tuning BERT to detect topic-agnostic media frames, achieving accuracy of 66.7% on the test set (1,138 news articles). It can predict the following 15 frames: (i) economy, (ii) capacity and resources, (iii) morality, (iv) fairness and equality, (v) legality, constitutionality and jurisprudence, (vi) policy prescription and evaluation, (vii) crime and punishment, (viii) security and defense, (ix) health and safety, (x) quality of life, (xi) cultural identity, (xii) public opinion, (xiii) politics, (xiv) external regulation and reputation, and (xv) other.

5 Results and Discussion

Below, we present the results of our analysis.

5.1 Disinformation Analysis

Figure 4 shows a detailed distribution for each question. We can see that (i) most tweets contain a verifiable factual claim, (ii) about half of the tweets are factually true, (iii) most of them are of general interest to the public, (iv) about half of the tweets are not harmful to the society, to a person, a company, or a product, (v) many tweets are trivial to fact-check, (vi) some tweets spread rumors, panic, or make a joke.
5.2 Propaganda Analysis

**Propaganda** Figure 5 shows the results for the propaganda analysis of tweets associated with check-worthiness and harmfulness. We can see that check-worthy tweets are more propagandistic (right-side bars in Figure 5a). A large portion of them (left-side bars) are neither check-worthy nor propagandistic. On Figure 5b, we can see that harmful tweets (i.e., such spreading rumors, conspiracy, and panic) are (somewhat) more propagandistic than non-harmful ones.

Figure 5c shows the propaganda analysis for articles posted in Facebook groups. We can see that skeptical articles are more propagandistic.

**Propaganda Techniques** A more fine-grained analysis is important in order to understand the type of content that is shared/posted in social media. Thus, we analyzed tweets by categorizing them using propaganda techniques. Figure 6 shows a propaganda technique analysis for the tweets, which are also labeled for check-worthiness and harmfulness.
5.3 Framing

Our analysis of framing in tweets shows that *economy* is the dominant perspective, *health and safety* come second, and *legality* is third. Figure 7 reports the distribution of tweets manually annotated for check-worthiness and harmfulness and automatically analyzed for framing. Figure 7a shows that the most frequent check-worthy tweets are associated with *health, legality, crime and punishment*, whereas non-check-worthy are associated with *economy, politics, and quality of life*. Figure 7b reports the distribution of the framing and the harmfulness labels. Frames labeled as *economy* are non-harmful; *cultural identity, crime and punishment* are associated with rumor/conspiracy, while *health and safety* frames show panic.

Figure 7c reports the distribution of articles manually categorized as skeptical vs. concerned and automatically analyzed for framing. The plot shows that skeletal articles are associated with *quality of life, policy, legality, economy*, and *politics*, whereas concerned articles are associated with *health and safety*, and *cultural identity*.

6 Limitations

**Manual annotations** Our manual annotation for disinformation in tweets shows moderate to substantial agreement across the questions. We believe that this is reasonable given the complexity of the task.

**Automatic analysis** The performance of the automatic analysis varies across the different tasks (e.g., for propaganda analysis vs. framing), which can introduce noise in the results.

**Translation** We needed to translate the text from Bulgarian to English, which can add noise in case of translation errors. Although we performed a qualitative analysis on a sample of propaganda annotations and we found a good quality for our model’s predictions, in future work, we would like to train a model directly for Bulgarian.

7 Conclusion and Future Work

We presented our analysis of COVID-19 in Bulgarian social media with focus on tweets and on news articles posted in Facebook groups, which we collected in different time frames starting from January till November 2020. Then, we manually and automatically analyzed them using different aspects of disinformation, propaganda, and framing.
Figure 6: Propaganda techniques.

We believe that the kind of analysis we perform here would help in better understanding various trends in social media about COVID-19. See also a related study about COVID-19 and vaccines in Qatar (Nakov et al., 2021a).

There are a number of interesting research directions that could be pursued using the approaches we used in this study. While we only focused on Twitter and Facebook, similar analysis can be done on other platforms e.g., WhatsApp, Gab, Reddit.
Figure 7: Framing.

Acknowledgments

We would like to thank the Atlantic Club in Bulgaria and DataBee for helping us with the manual analysis of Bulgarian tweets and of news articles posted on Facebook. We further thank the anonymous reviewers for their constructive comments.

This research is part of the Tanbih mega-project (http://tanbih.qcri.org), developed at the Qatar Computing Research Institute, HBKU, which aims to limit the impact of “fake news”, propaganda, and media bias by making users aware of what they are reading, thus promoting media literacy and critical thinking.
References


Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2016. A survey on truth discovery. SIGKDD Explor. Newsl., 17(2):1–16.


A Second Pandemic?
Analysis of Fake News About COVID-19 Vaccines in Qatar

Preslav Nakov, 1 Firoj Alam, 1 Shaden Shaar, 1
Giovanni Da San Martino 2 and Yifan Zhang 1
1 Qatar Computing Research Institute, HBKU, Qatar
2 University of Padova, Italy
{pnakov, fialam, sshaar, yzhang}@hbku.edu.qa,
dasan@math.unipd.it

Abstract

While COVID-19 vaccines are finally becoming widely available, a second pandemic that revolves around the circulation of anti-vaxxer “fake news” may hinder efforts to recover from the first one. With this in mind, we performed an extensive analysis of Arabic and English tweets about COVID-19 vaccines, with focus on messages originating from Qatar. We found that Arabic tweets contain a lot of false information and rumors, while English tweets are mostly factual. However, English tweets are much more propagandistic than Arabic ones. In terms of propaganda techniques, about half of the Arabic tweets express doubt, and 1/5 use loaded language, while English tweets are abundant in loaded language, exaggeration, fear, name-calling, doubt, and flag-waving. Finally, in terms of framing, Arabic tweets adopt a health and safety perspective, while in English economic concerns dominate.

1 Introduction

During the COVID-19 pandemic, social media have become one of the main communication channels for information dissemination and consumption, and many people rely on them as their primary source of news (Perrin, 2015), attracted by the broader choice of information sources. Unfortunately, over time, social media have also become one of the main channels to spread disinformation. To tackle this issue, a number of (mostly manual) fact-checking initiatives have been launched, and there are over 200 fact-checking organizations currently active worldwide. 1 However, these efforts are insufficient, given the scale of disinformation, which, in the time of COVID-19, has grown into the First Global Infodemic (according to the World Health Organization).

Figure 1: Tweets about COVID-19 and vaccines.

(a) rumor in Arabic and English

Military helicopters spray pesticides against Coronavirus. This is rumor!

(b) action taken

Qatar has signed a deal with @moderna_tx to buy its COVID-19 #vaccine when it is approved and released for global use.

(c) panic

Modernina COVID vaccine has caused side effect for those with cosmetic facial fillers foxnews.com/ue/moderna-cov... #FoxNews

(d) joke

I’ve just been informed that although the Covid vaccine won’t contain microchips, it will have the new U2 album

(e) bad cure

Nameste Twitter 😁

Today I wanna share something useful & effective for all of us 😊

GOOD NEWS!!! Wuhan’s Corona virus can cure itself by a bowl of freshly boiled garlic water. The old Chinese doctor proved its effectiveness. Many patients have also proven it to be effective.

Figure 1 shows examples of how online users discuss COVID-19 and vaccines. We can see that the problem goes beyond factuality: there are tweets spreading rumors, discussing action taken, instilling panic, making jokes, and promoting bad cure.

1 http://tiny.cc/zdlfnz
For the tweets in Figure 1, we might want to know whether they are factual, harmful, calling for action, etc. (see Section 4.1). It is also important to understand whether the content of the tweet is propagandistic (Section 4.2), what propaganda techniques are used, as well as the way the issue is framed (Section 4.3). Doing this in a timely manner is crucial to help organizations channel their efforts, and to counter the spread of disinformation, which may cause panic, mistrust, and other problems.

With this in mind, we performed an extensive analysis of Arabic and English tweets about COVID-19 and vaccines, with focus on messages originating from Qatar. Our analysis focuses on (i) COVID-19 disinformation, (ii) propaganda and its techniques, and (iii) framing.

Our contributions can be summarized as follows:

- We build and release a dataset of tweets related to COVID-19 and vaccines in Arabic and English.²
- We analyze the tweets from various perspectives (factuality, harmfulness, propaganda, and framing), and we discuss some interesting observations from our analysis.

2 Related Work

Below, we discuss relevant research directions.

2.1 Factuality

Work on fighting disinformation and misinformation online has focused on fact-checking and fake news detection (Li et al., 2016; Hardalov et al., 2016; Shu et al., 2017; Karadzhov et al., 2017; Lazer et al., 2018; Mishaylova et al., 2018; Vosoughi et al., 2018; Vo and Lee, 2018; Atanasova et al., 2019; Baly et al., 2019; Zlatkova et al., 2019; Baly et al., 2020; Nguyen et al., 2020; Shaar et al., 2020a). Research was further enabled by the emergence of datasets (Wang, 2017; Augenstein et al., 2019), often released as part of evaluation campaigns (Derczynski et al., 2017; Nakov et al., 2018; Da San Martinot et al., 2019; Elsayed et al., 2019; Gorrell et al., 2019; Mishaylova et al., 2019; Barrón-Cedeño et al., 2020; Nakov et al., 2021c,d; Shaar et al., 2021b). As automated systems have credibility issues (Arnold, 2020), another research direction has emerged: building tools to facilitate human fact-checkers (Nakov et al., 2021b).

2.2 Check-Worthiness Estimation

Given the volume of claims appearing in social media posts or in political statements, a problem that is crucial for fact-checkers is to identify which claims should be prioritized for fact-checking. The ClaimBuster system (Hassan et al., 2015) was a pioneering work in that direction. It categorized a political statement as non-factual, unimportant factual, or check-worthy factual. Gencheva et al. (2017) also focused on the 2016 US Presidential debates, for which they obtained binary (check-worthy vs. non-check-worthy) labels based on the fact-checking decisions of nine fact-checking organizations. An extension of this work was the ClaimRank system, which supports both English and Arabic (Jaradat et al., 2018). Note that political debates and speeches require modeling the context of the target sentence to classify. Indeed, context was a major focus for most research in the debates domain (Gencheva et al., 2017; Patwari et al., 2017; Vasileva et al., 2019; Shaar et al., 2021a). For example, Vasileva et al. (2019) modeled context in a multi-task learning neural network that predicts whether a sentence would be selected for fact-checking by each fact-checking organization (from a set of nine such organizations).

There has also been research on detecting check-worthy claims in social media (as opposed to the above research, which targeted political debates and speeches), featuring tweets about COVID-19 or general topics in Arabic and English (Hasanain et al., 2020; Shaar et al., 2020b, 2021c).

More directly related to our work here is the work of Alam et al. (2021c) and Alam et al. (2021a), who developed a multi-question annotation schema of tweets about COVID-19, organized around seven questions that aim to model the perspective of journalists, fact-checkers, social media platforms, policymakers, and the society. In our experiments, we use their schema and data to train classifiers for part of our analysis.

2.3 Propaganda

Propaganda is a communication tool that is deliberately designed to influence the opinions and the actions of other people in order to achieve a predetermined objective. When automatic means are being used to spread such influencing messages on social media platforms, this is referred to as computational propaganda (Woolley and Howard, 2018).

Research on propaganda detection has focused on textual content (Barrón-Cedeno et al., 2019; Rashkin et al., 2017; Da San Martino et al., 2019; Da San Martino et al., 2020a). Suitable datasets were made available by Rashkin et al. (2017) and Barrón-Cedeno et al. (2019), where the documents (news articles) were annotated using distant supervision, according to the reputation of their source, as judged by journalists. Rashkin et al. (2017) focused on analyzing the language of propaganda (vs. trusted, satire, and hoaxes) based on LIWC lexicons, while Barrón-Cedeno et al. (2019) studied a variety of stylistic features.

Habernal et al. (2017, 2018) developed a corpus annotated with five fallacies, including ad hominem, red herring, and irrelevant authority. Fine-grained propaganda analysis was done by Da San Martino et al. (2019), who developed a corpus of news articles annotated with 18 propaganda techniques. Subsequently, the Prta system was released (Da San Martino et al., 2020b), and improved models were proposed, focusing on interpretability (Yu et al., 2021) or addressing the limitations of transformers (Chernyavskiy et al., 2021). Finally, multimodal content was explored in memes using 22 propaganda techniques (Dimitrov et al., 2021a,b).

2.4 Framing

Framing refers to representing different salient aspects and perspectives for the purpose of conveying the latent meaning about an issue (Entman, 1993). Recent work on automatically identifying media frames includes developing coding schemes and semi-automated methods (Boydston et al., 2013), datasets such as the Media Frames Corpus (Card et al., 2015), and systems to automatically detect media frames (Liu et al., 2019; Zhang et al., 2019), large-scale automatic analysis of news articles (Kwak et al., 2020), and semi-supervised approaches (Cheeks et al., 2020).

2.5 Fighting the COVID-19 Infodemic

Related work on fighting the COVID-19 infodemic includes developing multi-question annotation schemes of tweets about COVID-19 (Alam et al., 2021c,a), studying credibility (Cinelli et al., 2020; Pulido et al., 2020; Zhou et al., 2020), racial prejudices and fear (Medford et al., 2020; Vidgen et al., 2020), situational information, e.g., caution and advice (Li et al., 2020), as well as detecting mentions and stance with respect to known misconceptions (Hossain et al., 2020).

3 Dataset

We collected Arabic tweets from February 2020 till March 2021. For the English tweets, we had two separate time periods (before and after COVID-19 vaccines became available): (i) from February till August 2020 (644 tweets), and (ii) from November 2020 till January 2021 (1,945 tweets). We used the following keywords to collect the tweets:

**English:** #covid19, #CoronavirusOutbreak, #Coronavirus, #Corona, #CoronaAlert, #CoronaOutbreak, Corona, covid-19, COVID vaccine, Covid-19 vaccine, #covidvaccine, corona vaccine, #vaccinate, #vaccine, vaccine

**Arabic:** كورونا, كورونا الجديد, #فيروس كورونا المستجد, #فيروس كورونا, #فیروس کورونا, #فیروس کورونا, #فیروس کورونا, #کورونا, #کورونا الجدید, لقاء, معلومات, تطعيم, لقاءات.

We collected original tweets (no retweets or replies), we removed the duplicates using the similarity-based approach in Alam et al. (2021b), and we filtered out tweets with less than five words. Finally, we kept the most frequently liked and retweeted tweets for annotation. Our final corpus consists of 606 Arabic and 2,589 English tweets.

4 Method

Figure 2 shows the architecture of our system. Below, we discuss each analysis step in detail.

4.1 Disinformation Analysis

For disinformation analysis, we used the dataset from (Alam et al., 2021c,a), which is organized around seven questions: asking whether the tweet (Q1) contains a verifiable factual claim, (Q2) is likely to contain false information, (Q3) is of interest to the general public, (Q4) is potentially harmful to a person, a company, a product, or the society, (Q5) requires verification by a fact-checker, (Q6) poses harm to society, or (Q7) requires the attention of policy makers. The dataset consist of 504 English and 218 Arabic tweets, and we used it to train an SVM classifier, whose hyper-parameters we optimized using 10-fold cross-validation. Table 1 shows the performance of the classifier for English and Arabic for all questions. Note the multiclass nature of the tasks and the skewed class distribution for Q2 to Q6 (Alam et al., 2021a).
The **Prta system** offers a fragment-level and a sentence-level classifiers. They were trained on a corpus of 350K tokens. The performance of the sentence-level classifier is 60.71 in terms of F1 score. The fragment-level classifier identifies the text fragments and the propaganda techniques that occur in them. They consider the following 18 techniques: (i) Loaded language, (ii) Name calling or labeling, (iii) Repetition, (iv) Exaggeration or minimization, (v) Doubt, (vi) Appeal to fear/prejudice, (vii) Flag-waving, (viii) Causal oversimplification, (ix) Slogans, (x) Appeal to authority, (xi) Black-and-white fallacy, dictatorship, (xii) Thought-terminating cliché, (xiii) Whataboutism, (xiv) Reductio ad Hitlerum, (xv) Red herring, (xvi) Bandwagon, (xvii) Obfuscation, intentional vagueness, confusion, and (xviii) Straw man.

Note that both Proppy and Prta were developed for English. Thus, for the classification of Arabic content, we first translated it to English using the Google translation API, and then we ran the tools.

### 4.3 Framing

We used the **Tanbih Framing Bias Detection system** (Zhang et al., 2019), trained on the Media Frames Corpus (11k training news articles) by fine-tuning BERT to detect topic-agnostic media frames, achieving accuracy of 66.7% on the test set (1,138 news articles). It can predict the following 15 frames: (i) Economy, (ii) Capacity and resources, (iii) Morality, (iv) Fairness and equality, (v) Legality, constitutionality and jurisprudence, (vi) Policy prescription and evaluation, (vii) Crime and punishment, (viii) Security and defense, (ix) Health and safety, (x) Quality of life, (xi) Cultural identity, (xii) Public opinion, (xiii) Politics, (xiv) External regulation and reputation, and (xv) Other.

### 5 Results and Discussion

#### 5.1 Disinformation Analysis

**Arabic:** Figure 3 shows the distribution for the questions for Arabic. We can see that (i) most tweets contain a verifiable factual claim, (ii) about half of the tweets contain false information, (iii) most tweets are of general interest to the public, (iv) about half of the tweets are harmful to the society, a person, a company, or a product (Question 6), (v) many tweets are worth fact-checking, (vi) most tweets are not harmful to the society, and many spread rumors, and (vii) some tweets discuss possible cure, and very few spread panic.
English: Figure 4 shows the distribution for the English tweets from February till August 2020. We can see that most tweets contain a verifiable factual claim, contain no false information, are of general interest to the public, are not harmful, and are worth fact-checking. Moreover, many tweets contain jokes, some contain rumors, and some blame the authorities.

We also analyzed the English tweets from November 2020 till January 2021. The results are shown in Figure 5, and follow a very similar trend.

Summary: Arabic tweets contain relatively more false information and rumors, some discuss possible cure, and very rarely spread panic. English tweets contain mostly factual statements, many make jokes, and rarely spread rumors.

5.2 Propaganda Analysis

Figure 6 shows the propaganda analysis in Arabic vs. English tweets. We can see that Arabic propagandistic tweets are extremely rare, while for English they about 33% of all tweets.
We also analyzed English tweets collected from November 2020 till January 2021, to cover tweets about COVID-19 vaccines, and we found that there were fewer propagandistic tweets: about 25%.

**Fine-Grained Propaganda Analysis** Next, we aimed to detect the specific propaganda techniques used in the tweets. Figure 7 shows the top propaganda techniques for Arabic and English.

We can see that, for Arabic, 50% of the tweets express doubt, and 20% use loaded language.

For English, we see a different distribution: about 33% of the tweets use loaded language, while each of the following techniques appears in about 10% of the tweets: exaggeration, fear, name-calling, doubt, and flag-waving.

Yet another trend is observed for English tweets collected from November 2020 till January 2021 (discussing vaccines): 50% of the tweets use loaded language, and each of the following four techniques appears in about 10% of the tweets: flag-waving, name-calling, and exaggeration.
5.3 Framing

Finally, we performed analysis in terms of framing, which reflects the perspective taken in the COVID-19 related tweets we analyzed. The results are shown on Figure 8.

We can see that in the Arabic tweets health and safety is the dominant frame, with economy coming second, and cultural identity being third.

For English, in both studied time periods, economy is the primary frame, and health and safety comes second.

We speculate that the difference in framing between Arabic and English tweets reflects the perspective of Qatari locals (who tweet primarily in Arabic) vs. that of expats (who tweet primarily in English). Thus, it is to be expected that the former are concerned primarily with health aspects (e.g., COVID-19 vaccination, social distancing, and other measures to keep one safe during the pandemic), while the latter worry more about the economic consequences of the pandemic (and respectively, about the security of their jobs).
6 Conclusion and Future Work

We have presented our analysis of COVID-19 tweets in Arabic and English aiming to help in the fight against the global infodemic, which emerged as a result of the COVID-19 pandemic. In particular, we collected tweets in different time frames starting from February 2020 till January 2021, and we analyzed them using different aspects of disinformation, propaganda, and framing. We believe that such analysis should help in better understanding the trends over time and across languages.

Many interesting directions could be pursued in future work. For example, the analysis could be applied to other languages; in fact, we already did a related study for Bulgarian (Nakov et al., 2021a). Moreover, while here we focused on tweets, the approach is applicable to other social media platforms such as Facebook and WhatsApp.

Acknowledgments

This research is part of the Tanbih mega-project,\(^3\) developed at the Qatar Computing Research Institute, HBKU, which aims to limit the impact of “fake news”, propaganda, and media bias by making users aware of what they are reading, thus promoting media literacy and critical thinking.

\(^3\)http://tanbih.qcri.org
References


Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2016. A survey on truth discovery. SIGKDD Explor. Newsl., 17(2):1–16.


A Hierarchical Entity Graph Convolutional Network for Relation Extraction across Documents

Tapas Nayak∗
Department of Computer Science
Indian Institute of Technology Kharagpur
tnk02.05@gmail.com

Hwee Tou Ng
Department of Computer Science
National University of Singapore
nght@comp.nus.edu.sg

Abstract

Distantly supervised datasets for relation extraction mostly focus on sentence-level extraction, and they cover very few relations. In this work, we propose cross-document relation extraction, where the two entities of a relation tuple appear in two different documents that are connected via a chain of common entities. Following this idea, we create a dataset for two-hop relation extraction, where each chain contains exactly two documents. Our proposed dataset covers a higher number of relations than the publicly available sentence-level datasets. We also propose a hierarchical entity graph convolutional network (HEGCN) model for this task that improves performance by 1.1% F1 score on our two-hop relation extraction dataset, compared to some strong neural baselines.

1 Introduction

The idea of distant supervision (Mintz et al., 2009) eliminates the need for manual annotation for obtaining training data for relation extraction. Previously, this idea is used mostly to create sentence-level datasets. However, the assumption of distant supervision, that the two entities of a tuple must appear in the same sentence, is overly strict. We may not find an adequate number of evidence sentences for many relations as both entities do not appear in the same sentence. The relation extraction models built on such data can find relations only for a small number of relations and the relations of most knowledge bases (KBs) will be out of the reach of such models.

To address this issue, we propose a multi-hop relation extraction task where the subject and object entities of a tuple can appear in two different documents, and these two documents are connected via some common entities. We can create a chain of entities from the subject entity to the object entity of a tuple via the common entities across multiple documents. Each link in this chain represents a relation between the entities located at the endpoints of the link. We can determine the relation between the subject and object entities of a tuple by following this chain of relations. This approach can give training instances for more relations than sentence-level distant supervision. Following the proposed multi-hop approach, we create a two-hop relation extraction dataset for the task. Each instance of this dataset has two documents, where the first document contains the subject entity and the second document contains the object entity of a tuple. These two documents are connected via at least one common entity. This idea can be extended to create an N-hop dataset.

We also propose a hierarchical entity graph convolutional network (HEGCN) model for the task. Our proposed model has two levels of graph convolutional networks (GCNs). The first-level GCN of the hierarchy is applied to the entity mention level graph of every document to capture the relations among the entity mentions within a document. The second-level GCN of the hierarchy is applied on a unified entity-level graph, which is built using all the unique entities present in the document chain. This entity-level graph can be built on the document chain of any length and it can capture the relations among the entities across the multiple documents in the chain. Our proposed HEGCN model improves the performance on our two-hop dataset. To summarize, the following are the contributions of this paper:

(1) We propose a multi-hop relation extraction task and create a two-hop dataset. This dataset has more relations than other popular distantly supervised sentence-level or document-level relation extraction datasets.

∗ This work was done when the first author was a PhD student at the National University of Singapore.
(2) We propose a novel hierarchical entity graph convolutional network (HEGCN) for multi-hop relation extraction. Our proposed model improves the F1 score by 1.1% on our two-hop dataset, compared to strong neural baselines.\footnote{The source code and data for this paper are available at https://github.com/nusnlp/MHRE.git}

2 Task Formalization

Multi-hop relation extraction can be defined as follows. Consider two entities, a subject entity $e_s$ and an object entity $e_o$, and a chain of documents $D = \{D_s \rightarrow D_1 \rightarrow D_2 \rightarrow \ldots \rightarrow D_n \rightarrow D_o\}$ where $e_s \in D_s$ and $e_o \in D_o$. There exists a chain of entities $e_s \rightarrow c_1 \rightarrow c_2 \rightarrow \ldots \rightarrow c_{n+1} \rightarrow e_o$ where $c_1 \in \{D_s, D_1\}, c_2 \in \{D_1, D_2\}, \ldots, c_{n+1} \in \{D_n, D_o\}$. The task is to find the relation between $e_s$ and $e_o$ from a pre-defined set of relations $R \cup \{\text{None}\}$, where $R$ is the set of relations and None indicates that none of the relations in $R$ holds between $e_s$ and $e_o$. A simpler version of this task is two-hop relation extraction where $D_s$ and $D_o$ are directly connected by at least one common entity. In this paper, we focus on two-hop relation extraction.

3 Related Work

3.1 Relation Extraction Datasets

Distantly supervised datasets are very popular for relation extraction (Nayak et al., 2021). Riedel et al. (2010) (NYT10) and Hoffmann et al. (2011) (NYT11) mapped Freebase tuples to New York Times (NYT) articles to obtain such datasets. The NYT10 and NYT11 datasets have been used extensively by researchers for relation extraction. TACRED (Zhang et al., 2017) is another dataset created from the TAC KBP evaluations. FewRel 2.0 (Gao et al., 2019) is a few-shot relation extraction dataset. All these datasets are created at the sentence level. DocRED (Yao et al., 2019) is a document-level relation extraction dataset created using Wikipedia articles and Wikidata items. To the best of our knowledge, there does not exist any relation extraction dataset which involves multiple documents.

3.2 Relation Extraction Models

Neural models have performed well on distantly supervised datasets for relation extraction. Zeng et al. (2014, 2015) used convolutional network with max-pooling on word embeddings for this task, whereas Shen and Huang (2016); Jat et al. (2017); Nayak and Ng (2019) used word-level attention model for single-instance sentence-level relation extraction. Lin et al. (2016); Vashishth et al. (2018); Ye and Ling (2019) used neural networks in a multi-instance setting to find a relation from a bag of independent sentences. Recently, graph convolutional network-based (GCN) (Kipf and Welling, 2017) models have become popular for many NLP tasks. These models work on non-linear graph structures. Zhang et al. (2018); Vashishth et al. (2018); Guo et al. (2019); Zeng et al. (2020) used graph convolution networks for relation extraction. They consider each token in a sentence as a node in the graph and use a syntactic dependency tree to create a graph structure among the nodes. Recently, neural joint extraction approaches (Takanobu et al., 2019; Nayak and Ng, 2020) were proposed for this task.

3.3 Multi-hop QA versus Multi-hop RE

Welbl et al. (2018) proposed a multi-hop QA dataset (WikiHop) where the answer can only be found using more than one document. Several neural models have been proposed (Song et al., 2018; Cao et al., 2019; De Cao et al., 2019; Kundu et al., 2019) to solve this task. We have created a two-hop relation extraction dataset (THRED) from this WikiHop dataset. The major difference between these two datasets is that THRED contains many None relations, whereas in the WikiHop dataset, every instance has a correct answer. Extracting the None relation is challenging, since None occurs when no relations in $R$ exist. When the number of relations in $R$ increases, it becomes more difficult to predict the relations. As such, we believe the multi-hop RE task is more challenging than the multi-hop QA task.

4 Dataset Construction

We create a two-hop relation extraction dataset from a multi-hop question-answering (QA) dataset WikiHop (Welbl et al., 2018). Welbl et al. (2018) defined the multi-hop QA task as follows: Given a set of supporting documents $D_s$ and a set of candidate answers $C_a$ which are mentioned in $D_s$, the goal is to find the correct answer $a^* \in C_a$ for a question by drawing on the supporting documents. They used Wikipedia articles and Wikidata (Vrandečić and Krötzsch, 2014) tuples for creating this dataset. Each positive tuple $(e_s, e_o, r_p)$ in
WikiHop test data is blind and not released. We find that 76% of the selected positive samples and 82% of the selected negative samples are accurate.

### 4.1 Dataset Statistics

The training, validation, and test data of the WikiHop dataset are created using distant supervision, but the validation and test data are manually verified. WikiHop test data is blind and not released. We use their validation data to create the test data for our task and use their training data for our training and validation purposes. We include the statistics of our two-hop relation extraction dataset.

\[ (e_s, a^*, r_p) \]

is the positive tuple for relation extraction. For any other candidate answer \( e_w \in C_a - \{a^*\} \), the entity pair \((e_s, e_w)\) is considered as a None tuple if there exists no relation among the four pairs \((e_s, e_w), (e_w, e_s), (e_w, e_o)\), and \((e_o, e_w)\) in Wikidata. We check for the no relation condition for these four entity pairs involving \(e_w, e_s, e_o\) to reduce the distant supervision noise in the dataset for None tuples. We create a None candidate set \(C_n\) with each \(e_w \in C_a - \{a^*\}\). We first find all possible pairs of documents from the supporting document set \(D_s\) such that the first document of the pair contains the subject entity \(e_s\) and the second document of the pair contains either the entity \(a^*\) or one of the entities from \(C_n\). We discard those pairs of documents that do not contain any common entity. The document pairs where the second document contains the entity \(a^*\) are considered as a document chain for the positive tuple \((e_s, a^*, r_p)\) where \(r_p \in R\). All other document pairs where the second document contains an entity from the set \(C_n\) are considered as a document chain for None tuple \((e_s, e_w, None)\) where \(e_w \in C_n\). In this way, using distant supervision, we can create a dataset for two-hop relation extraction. Each instance of this dataset has a chain of documents \(D = \{D_s \rightarrow D_o\}\) of length 2 that is the textual source of a tuple \((e_s, e_o, r)\). The document \(D_s\) contains the subject entity \(e_s\) and the document \(D_o\) contains the object entity \(e_o\). The two documents are connected with at least one common entity \(c\). There exists at least one entity chain \(e_s \rightarrow c \rightarrow e_o\) in the document chain. The goal is to find the relation \(r\) between \(e_s\) and \(e_o\) from the set \(R \cup \{None\}\). We refer to this two-hop dataset as THRED (two-hop relation extraction dataset) in the remaining sections of this paper. We manually checked 100 randomly selected positive samples and 100 randomly selected negative samples, and found that 76% of the selected positive samples and 82% of the selected negative samples are accurate.

\[ D = \{D_s \rightarrow D_o\}\]
Table 1: A multi-hop question-answer instance from the WikiHop dataset. The tuple (Doc1, Zoo Lake, Doc2, Gauteng, located_in_administrative_entity) constitutes a positive instance in the THRED dataset. The tuple (Doc1, Zoo Lake, Doc3, Tanzania, None) constitutes a negative instance in the THRED dataset.

<table>
<thead>
<tr>
<th>#Common entities</th>
<th>#Document chains</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
</tr>
<tr>
<td>1</td>
<td>92,140</td>
</tr>
<tr>
<td>2</td>
<td>36,275</td>
</tr>
<tr>
<td>3</td>
<td>10,824</td>
</tr>
<tr>
<td>4</td>
<td>3,170</td>
</tr>
<tr>
<td>≥5</td>
<td>1,497</td>
</tr>
</tbody>
</table>

Table 4: The number of relations in various relation extraction datasets. $R$ is the set of positive relations.

5 Proposed HEGCN Model

We propose a hierarchical entity graph convolutional network (HEGCN) for multi-hop relation extraction. We encode the documents in a document chain using a bi-directional long short-term memory (BiLSTM) layer (Hochreiter and Schmidhuber, 1997). On top of the BiLSTM layer, we use two graph convolutional networks (GCN), one after another in a hierarchy. In the first level of the GCN hierarchy, we construct a separate entity mention graph on each document of the chain using all the entities mentioned in that document. Each mention of an entity in a document is considered as a separate node in the graph. We use a graph convolutional network (GCN) to represent the entity mention graph of each document to capture the relations among the entity mentions in the document. We then construct a unified entity-level graph across all the documents in the chain. Each node of this entity-level graph represents a unique entity in the document chain. Each common entity between two documents in the chain is represented by a single node in the graph. We use a GCN to represent this entity-level graph to capture the relations among the entities across the documents. We concatenate the representations of the nodes of the subject entity and object entity and pass it to a feed-forward layer with softmax for relation classification.

5.1 Documents Encoding Layer

We use two types of embedding vectors: (1) word embedding vector $w \in \mathbb{R}^{d_w}$ (2) entity token indicator embedding vector $z \in \mathbb{R}^{d_z}$, which indicates if a word belongs to the subject entity, object entity, or common entities. The subject and object entities are assigned the embedding index of 2 and 3, respectively. The common entities in the document chain are assigned embedding index in an...
increasing order starting from index 4. The same entities present in two documents in the chain get the same embedding index. Embedding index 0 is used for padding and 1 is used for all other tokens in the documents. A document is represented using a sequence of vectors \( \{x_1, x_2, \ldots, x_n\} \) where \( x_t = w_t||z_t. \) \( || \) represents the concatenation of vectors and \( n \) is the document length. We concatenate all documents in a chain sequentially by using a document separator token. These token vectors are passed to a BiLSTM layer to capture the interaction among the documents in a chain. \( \tilde{h}_t \in \mathbb{R}^{(d_w+d_z)} \) and \( \tilde{h}_t \in \mathbb{R}^{(d_w+d_z)} \) are the output at the \( t \)th step of the forward LSTM and backward LSTM respectively. We concatenate them to obtain the \( t \)th BiLSTM output \( h_t \in \mathbb{R}^{2(d_w+d_z)} \).

5.2 Hierarchical Entity Graph Convolutional Layers

Kipf and Welling (2017) proposed graph convolutional networks (GCN) which work on graph structures. Here, we describe the GCN which is used in our model. We represent a graph \( G \) with \( m \) nodes using an adjacency matrix \( A \) of size \( m \times m \). If there is an edge between node \( i \) and node \( j \), then \( A_{ij} = A_{ji} = 1 \). We also add self loops, \( A_{ii} = 1 \), in the graph \( G \). We normalize the adjacency matrix \( A \) by using symmetric normalization proposed by Kipf and Welling (2017). A diagonal node degree matrix \( D \) of size \( m \times m \) is used in the normalization of \( A \). \( \text{deg}(v_i) \) is the number of edges that are connected to the node \( v_i \) in \( G \) and \( \hat{A} \) is the corresponding normalized adjacency matrix of \( G \). Each node of the graph receives the hidden representation of its neighboring nodes from the \( (l-1) \)th layer and uses the following operation to update its own hidden representation.

\[
D_{ij}^{-\frac{1}{2}} = \begin{cases} \frac{1}{\sqrt{\text{deg}(v_i)}} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}
\]

\[
\hat{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}
\]

\[
g_l^i = \text{ReLU}(\sum_{j=1}^m \hat{A}_{ij} W_l^i g_{j, l-1}^j)
\]

\( W_l \) is the trainable weight matrix of the \( l \)th layer of the GCN. \( g_l^i \) is the representation of the \( i \)th node of the graph at the \( l \)th layer. If \( g_l^i \) has the dimension of \( d_g \), then the dimension of the weight matrix \( W_l \) is \( d_g \times d_g \). \( g_0^i \) is the initial input to the GCN.

5.2.1 Entity Mention Graph Layer

We construct an entity mention graph (EMG) for each document in the chain on top of the document encoding layer. An entity string may appear at multiple locations in a document and each appearance is considered as an entity mention. We add a node in the graph for each entity mention. We connect two entity mention nodes if they appear in the same sentence (EMG type 1 edge). We assume that since they appear in the same sentence, there may exist some relation between them. We also connect two entity mention nodes if the strings of the two entity mentions are identical (EMG type 2 edge). Let \( e_1, \ldots, e_l \) be the sequence of entity mention nodes listed in the order of their appearance in a document. We connect nodes \( e_i \) and \( e_{i+1} \) \((1 \leq i < l)\) with an edge (EMG type 3 edge). EMG type 3 edges create a linear chain of the entity mentions and ensure that the graph is connected. We
use a graph convolutional network on this graph topology to capture the relations among the entity mentions in a document.

We obtain the initial representations of the entity mention nodes from the hidden representations of the document encoding layer. We concatenate the hidden vector of the first token of an entity mention, the hidden vector of its last token, and a context vector to obtain the entity mention node representation. The context vector is obtained using an attention mechanism on the tokens of the sentence in which the entity mention appears.

\[ p = h_b || h_e, \quad s_t = \tanh(p^T W)h_t \]

\[ a = \text{softmax}([s_1, s_2, \ldots, s_k]^T) \]

\[ c = \sum_{t=1}^{k} a_t h_t, \quad q = p || c \]

\[ h_b \in \mathbb{R}^{2(d_w+d_z)} \text{ and } h_e \in \mathbb{R}^{2(d_w+d_z)} \] are the hidden vectors from the document encoding layer of the first and last token of an entity mention. \( W \in \mathbb{R}^{4(d_w+d_z) \times 2(d_w+d_z)} \) is a trainable weight matrix, \( h_t \in \mathbb{R}^{2(d_w+d_z)} \) is the hidden vector of the \( t \)th token of the sentence in which the entity mention is located, and \( a_t \) is the normalized attention score for the \( t \)th token with respect to the entity mention. \( k \) is the length of the sentence in which the entity mention is located, and \( c \in \mathbb{R}^{2(d_w+d_z)} \) is the context vector. The entity mention node vector \( q \in \mathbb{R}^{2(d_w+d_z)} \) of the \( i \)th node in the graph is passed to the GCN as \( g_i^0 \). The parameters of this GCN are shared across the documents in a chain. This layer of the model is referred to as entity mention-level graph convolutional network or EMGCN.

### 5.2.2 Entity Graph Layer

We construct a unified entity graph (EG) on top of the entity mention graphs. First, we construct an entity graph for each document, where each unique entity string is represented as an entity node in the graph. We add an edge between two entity nodes if the strings of the two entities appear together in at least one sentence in the document (EG type 1 edge). We also form a sequence of entity nodes based on the order of appearance of the entities in a document, where only the first occurrence of multiple occurrences of an entity is kept in the sequence. We connect two consecutive entity nodes in the sequence with an edge (EG type 2 edge). This ensures that the entire entity graph remains connected.

We construct one entity graph for each document in the document chain. We unify the entity graphs of multiple documents by merging the nodes of common entities between them. The unified entity graph contains all the nodes from the multiple entity graphs, but the common entity nodes which appear in two entity graphs are merged into one node in the unified graph. There is an edge between two entity nodes in the unified entity graph if there exists an edge between them in any of the entity graphs of the documents.

We obtain the initial representations of the entity nodes from the GCN outputs of the entity mention graphs. For the common entities between two documents, we average the GCN outputs of the entity mention nodes that have an identical string as the entity from the entity mention graphs of the two documents. For other entity nodes that appear only in one document, we average the GCN outputs of the entity mention nodes that have an identical string as the entity from the entity mention graph of that document. Each entity vector is passed to another graph convolutional network as \( g_i^1 \) which represents the initial representation of the \( i \)th entity node in the unified entity graph. We use a graph convolutional network on this graph topology to capture the relations among the entities across the documents in the document chain. This layer of the model is referred to as entity level graph convolutional network or EGCGN.

### 5.3 Relation Classifier

We concatenate the EGCGN outputs of the nodes corresponding to the subject entity \( e_s \in \mathbb{R}^{6(d_w+d_z)} \) and object entity \( e_o \in \mathbb{R}^{6(d_w+d_z)} \), and pass the concatenated vector to a feed-forward network (FFN) with softmax to predict the normalized probabilities for the relation labels.

\[ r = \text{softmax}(W_r(e_s || e_o) + b_r) \]

\( W_r \in \mathbb{R}^{(|R|+1) \times 2(d_w+d_z)} \) is the weight matrix, \( b_r \in \mathbb{R}^{|R|+1} \) is the bias vector of the FFN, and \( r \) is the vector of normalized probabilities of relation labels.

### 6 Experiments

#### 6.1 Baselines

We implement four neural baseline models for comparison with our proposed HEGCN model. Similar to our proposed model, we represent the tokens in the documents using pre-trained word embedding
vectors and entity token indicator vectors. We use a document separator token when concatenating the vectors of two documents in a chain.

(1) CNN: We apply the convolution operation on the sequence of token vectors with different kernel sizes. A max-pooling operation is applied to choose the features from the outputs of the convolution operation. This feature vector is passed to a feed-forward layer with softmax to classify the relation.

(2) BiLSTM: The token vectors of the document chain are passed to a BiLSTM layer to encode its meaning. We obtain the entity mention vectors of the subject entity and the object entity by concatenating the hidden vectors of their first and last token. We average the entity mention tokens of the corresponding entity to obtain the representation of the subject entity and the object entity. These two vectors are concatenated and passed to a feed-forward layer with softmax to find the relation between them.

(3) BiLSTM-CNN: This is a combination of the BiLSTM and CNN model described above. The token vectors of the documents are passed to a BiLSTM layer and then we use the convolution operation with max-pooling with different convolutional kernel sizes on the hidden vectors of the BiLSTM layer. The feature vector obtained from the max-pooling operation is passed to a feed-forward layer with softmax to classify the relation.

(4) LinkPath: This model uses the explicit paths (Kundu et al., 2019) from the subject entity $e_s$ to the object entity $e_o$ via the common entities to find the relation. As we consider only two-hop relations, each path from $e_s$ to $e_o$ will be of the form $e_s \rightarrow c \rightarrow e_o$, where $c$ is a common entity. Since there can be multiple common entities between two documents and these common entities as well as the subject and object entities can appear multiple times in the two documents, there exist multiple paths from $e_s$ to $e_o$. Each path is formed with four entity mentions: (i) entity mentions of the subject entity and common entity in the first document. (ii) entity mentions of the common entity and object entity in the second document. We concatenate the BiLSTM hidden vectors of the start and end token of an entity mention to obtain its representation. Each path is constructed by concatenating all the four entity mentions of the path. This can be extended from two-hop to multi-hop relations by using a recurrent neural network that takes the path entity mentions as input, and outputs the hidden representation of the path. We average the vector representations of all the paths and pass it to a feed-forward layer with softmax to find the relation.

6.2 Parameter Settings
We use GloVe (Pennington et al., 2014) word embeddings of dimension $d_w$ which is set to 300 in
our experiments, and update the embeddings during training. We set the dimension $d_z$ to be 20 for the entity token indicator embedding vectors. The hidden vector dimension of the forward and backward LSTM is set at 320. The dimension of BiLSTM output is 640. We use 500 different convolution filters with kernel width of 3, 4, and 5 for feature extraction. We use one convolutional layer in both entity mention-level GCN and entity-level GCN in our final model. Dropout layers (Srivastava et al., 2014) are used in our network with a dropout rate of 0.5 to avoid overfitting. We train our models with a mini-batch size of 32 and use negative log-likelihood as our objective function. We optimize the network parameters using the Adagrad optimizer (Duchi et al., 2011). For evaluation, we use precision, recall, and F1 score. We do not include the None relation in the evaluation. A confidence threshold that achieves the highest F1 score on the validation dataset is used to decide if the relation of a test instance belongs to the set of relations $R$ or None.

6.3 Experimental Results

We include the median of five runs of the models on the THRED dataset in Table 5. We see that adding a BiLSTM in the document encoding layer improves the performance by close to 5% in F1 score. The BiLSTM, BiLSTM_CNN, and LinkPath models achieve similar F1 scores. When we add our proposed hierarchical entity graph convolutional layer on top of the BiLSTM layer, we get another 1.1% F1 score improvement over the next best BiLSTM model. We perform a statistical significance test using bootstrap resampling to compare each baseline and our HEGCN model, and have ascertained that the higher F1 score achieved by our model is statistically significant ($p < 0.001$).

<table>
<thead>
<tr>
<th>L1</th>
<th>L2</th>
<th>Prec.</th>
<th>Rec.</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.674</td>
<td>0.699</td>
<td>0.686</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.709</td>
<td>0.650</td>
<td>0.678</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.682</td>
<td>0.663</td>
<td>0.673</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.671</td>
<td>0.635</td>
<td>0.653</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.673</td>
<td>0.667</td>
<td>0.670</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.623</td>
<td>0.651</td>
<td>0.637</td>
</tr>
</tbody>
</table>

Table 6: The ablation study of the HEGCN model with different numbers of convolutional layers (L1 and L2) in EMGCN and EGCN.

In Table 7, we include the ablation study of the different types of edges in EMGCN and EGCN. Removing any type of edges reduces the F1 score.

<table>
<thead>
<tr>
<th>Model</th>
<th>Prec.</th>
<th>Rec.</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEGCN</td>
<td>0.674</td>
<td>0.699</td>
<td>0.686</td>
</tr>
<tr>
<td>– EMG type 1</td>
<td>0.679</td>
<td>0.689</td>
<td>0.684</td>
</tr>
<tr>
<td>– EMG type 2</td>
<td>0.698</td>
<td>0.662</td>
<td>0.680</td>
</tr>
<tr>
<td>– EMG type 3</td>
<td>0.666</td>
<td>0.693</td>
<td>0.679</td>
</tr>
<tr>
<td>– EG type 1</td>
<td>0.704</td>
<td>0.659</td>
<td>0.681</td>
</tr>
<tr>
<td>– EG type 2</td>
<td>0.674</td>
<td>0.691</td>
<td>0.683</td>
</tr>
</tbody>
</table>

Table 7: The ablation study of the different types of edges in our HEGCN model.

7 Conclusion

In this paper, we propose how the idea of distant supervision can be extended from sentence-level extraction to multi-hop extraction to cover more relations. We propose a general approach to create multi-hop relation extraction datasets. Following this approach, we create a two-hop relation extraction dataset that covers a higher number of relations from knowledge bases than other distantly supervised relation extraction datasets. We also propose a hierarchical entity graph convolutional network for this task. The two levels of GCN in our model help to capture the relation cues within documents and across documents. Our proposed model improves the F1 score by 1.1% on our two-hop dataset, compared to a strong neural baseline, and it can be readily extended to N-hop datasets.

References

Yu Cao, Meng Fang, and Dacheng Tao. 2019. BAG: Bi-directional attention entity graph convolutional network for multi-hop reasoning question answering. In NAACL.


Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2019. FewRel 2.0: Towards more challenging few-shot relation classification. In EMNLP and IJCNLP.


Sharmistha Jat, Siddhesh Khandelwal, and Partha Talukdar. 2017. Improving distantly supervised relation extraction using word and entity based attention. In AKBC.


Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In ACL.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In ACL and IJCNLP.


Tapas Nayak and Hwee Tou Ng. 2019. Effective attention modeling for neural relation extraction. In CoNLL.

Tapas Nayak and Hwee Tou Ng. 2020. Effective modeling of encoder-decoder architecture for joint entity and relation extraction. In AAAI.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In EMNLP.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In ECML and KDD.

Yatian Shen and Xuanjing Huang. 2016. Attention-based convolutional neural network for semantic relation extraction. In COLING.


Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. DocRED: A large-scale document-level relation extraction dataset. In ACL.

Zhi-Xiu Ye and Zhen-Hua Ling. 2019. Distant supervision relation extraction with intra-bag and inter-bag attentions. In NAACL.


Shuang Zeng, Runxin Xu, Baobao Chang, and Lei Li. 2020. Double graph based reasoning for document-level relation extraction. In EMNLP.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In EMNLP.

Improving Distantly Supervised Relation Extraction
with Self-Ensemble Noise Filtering

Tapas Nayak
IIT Kharagpur
India

Navonil Majumder
SUTD
Singapore
{tnk02.05,n.majumder.2009,soujanya.poria}@gmail.com

Soujanya Poria
SUTD
Singapore

Abstract

Distantly supervised models are very popular for relation extraction since we can obtain a large amount of training data using the distant supervision method without human annotation. In distant supervision, a sentence is considered as a source of a tuple if the sentence contains both entities of the tuple. However, this condition is too permissive and does not guarantee the presence of relevant relation-specific information in the sentence. As such, distantly supervised training data contains much noise which adversely affects the performance of the models. In this paper, we propose a self-ensemble filtering mechanism to filter out the noisy samples during the training process. We evaluate our proposed framework on the New York Times dataset which is obtained via distant supervision. Our experiments with multiple state-of-the-art neural relation extraction models show that our proposed filtering mechanism improves the robustness of the models and increases their F1 scores.

1 Introduction

The task of relation extraction is about finding relation or no relation between two entities. This is an important task to fill the gaps of existing knowledge bases (KB). Open information extraction (OpenIE) (Banko et al., 2007) is one way of extracting relations from text. They consider the verb in a sentence as the relation and then find the noun phrases located to the left and right of that verb as the entities. But this process has two serious problems: First, the same relation can appear in the text with many verb forms and OpenIE treats them as different relations. This leads to the duplication of relations in KB. Second, OpenIE treats any verbs in a sentence as a relation which can generate a large number of insignificant tuples which cannot be added to a KB.

Supervised relation extraction models, on the other hand, do not have these problems. But they require a large amount of annotated data which is difficult to get. Mintz et al. (2009), Riedel et al. (2010), and Hoffmann et al. (2011) used the idea of distant supervision to automatically obtain the training data to overcome this problem. The idea of distant supervision is that if a sentence contains both the entities of a tuple, it is chosen as a source sentence of this tuple. Although this process can generate some noisy training instances, it can give a significant amount of training data which can be used to build supervised models for this task. They map the tuples from existing KBs such as Freebase (Bollacker et al., 2008) to the text corpus such as Wikipedia articles (Mintz et al., 2009) or New York Times articles (Riedel et al., 2010; Hoffmann et al., 2011).

Based on distantly supervised training data, researchers have proposed many state-of-the-art models for relation extraction. Mintz et al. (2009), Riedel et al. (2010), and Hoffmann et al. (2011) proposed feature-based learning models and used entity tokens and their nearby tokens, their part-of-speech tags, and other linguistic features to train their models. Recently, many neural network-based models have been proposed to avoid feature engineering. Zeng et al. (2014) and Zeng et al. (2015) used convolutional neural networks (CNN) with max-pooling to find the relation between two given entities. Shen and Huang (2016), Jat et al. (2017), Nayak and Ng (2019) used attention framework in their neural models for this task.

But the distantly supervised data may contain many noisy samples. Sometimes sentences may contain the two entities of a positive tuple, but they may not contain the relation specific information. These kinds of sentences and entity pairs are considered as positive noisy samples. Another set of noisy samples comes from the way samples for
None relation are created. If a sentence contains two entities from the KB and there is no positive relation between these two entities in the KB, this sentence and entity pair is considered as a sample for None relation. But knowledge bases are not complete and many valid relations among the entities in the KBs are missing. So it may be possible that the sentence contains information about some positive relation between the two entities, but since the relation is not present in the KB, this sentence and entity pair is incorrectly considered as a sample for None relation. These kinds of sentences and entity pairs are considered as negative noisy samples.

We include examples of clean and noisy samples generated using distant supervision in Table 1. The KB contains many entities out of which four entities are Barack Obama, Hawaii, Karkuli, and West Bengal. Barack Obama and Hawaii have a birth_place relation between them. Karkuli and West Bengal are not connected with any relations in the KB. So we assume that there is no valid relation between these two entities. The sentence in the first sample contains the two entities Barack Obama and Hawaii, and it also contains information about Obama being born in Hawaii. So this sentence is a correct source for the tuple (Barack Obama, Hawaii, birth_place). So this is a positive clean sample. The sentence in the second sample contains the two entities, but it does not contain the information about Barack Obama being born in Hawaii. So it is a positive noisy sample. In the case of the third and fourth samples, according to distant supervision, they are considered as samples for None relation. But the sentence in the third sample contains the information for the relation located_in between Karkuli and West Bengal. So the third sample is a negative noisy sample. The fourth sample is an example of a negative clean sample.

The presence of the noisy samples in the distantly supervised data adversely affects the performance of the models. Our goal is to remove the noisy samples from the training process to make the models more robust for this task. We propose a self-ensemble based noisy samples filtering method for this purpose. Our framework identifies the noisy samples during the training and removes them from training data in the following iterations. This framework can be used with any supervised relation extraction model. We run experiments with several state-of-the-art neural models, namely Convolutional Neural Network (CNN) (Zeng et al., 2014), Piecewise Convolutional Neural Network (PCNN) (Zeng et al., 2015), Entity Attention (EA) (Shen and Huang, 2016), and Bi-GRU Word Attention (BGWA) (Jat et al., 2017) with the distantly supervised New York Times dataset (Hoffmann et al., 2011). Our framework improves the F1 score of these models by 2.1%, 1.1%, 2.1%, and 2.3% respectively.

Table 1: Examples of distantly supervised (DS) clean and noisy samples.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Entity 1</th>
<th>Entity 2</th>
<th>DS Relation</th>
<th>Actual Relation</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barack Obama was born in Hawaii.</td>
<td>Barack Obama</td>
<td>Hawaii</td>
<td>birth_place</td>
<td>birth_place</td>
<td>Clean</td>
</tr>
<tr>
<td>Barack Obama visited Hawaii.</td>
<td>Barack Obama</td>
<td>Hawaii</td>
<td>birth_place</td>
<td>None</td>
<td>Noisy</td>
</tr>
<tr>
<td>Suvendu Adhikari was born at Karkuli in Purba Medinipur in West Bengal.</td>
<td>Karkuli</td>
<td>West Bengal</td>
<td>None</td>
<td>located_in</td>
<td>Noisy</td>
</tr>
<tr>
<td>Suvendu Adhikari, transport minister of West Bengal, visited Karkuli.</td>
<td>Karkuli</td>
<td>West Bengal</td>
<td>None</td>
<td>None</td>
<td>Clean</td>
</tr>
</tbody>
</table>
datasets are used for training relation extraction models. But the presence of noisy samples negatively affects their performance. In this work, we try to identify these noisy samples during training and filter them out from the subsequent training process to improve the performance of the models.

3 Self-Ensemble Filtering Framework

Figure 1 shows our proposed self-ensemble filtering framework. This framework is inspired from the work by Nguyen et al. (2020). We start with clean and noisy samples and assume that all samples are clean. At the end of each iteration, we predict the labels of the entire training samples. Based on the predicted label and the label assigned by distant supervision, we decide to filter out a sample in the next iteration. After each iteration, we consider the entire training samples for the filtering process. The individual models at each iteration can be very sensitive to wrong labels, so in our training process, we maintain a self-ensemble version of the models which is a moving average of the models of previous iterations. We hypothesize that the predictions of the ensemble model are more stable than the individual models. So the predictions from the ensemble model are used to identify the noisy samples. These noisy samples are removed from the training samples of the next iteration. We consider the entire distantly supervised training data for prediction and filtering so that if a sample is filtered out wrongly in an iteration, it can be included again in the training data in the subsequent iteration.

3.1 Self-Ensemble Training

We use the student-teacher training mechanism proposed by Tarvainen and Valpola (2017) for our self-ensemble model learning. A student model can be any supervised learning model such as a neural network model. A teacher model is the clone of student model with same parameters. The weights of parameters of this teacher model is the exponential moving average of the weights of parameters of the student model. So this teacher model is the self-ensemble version of the student model. An additional consistency loss is used to maintain the consistency of the predictions of the student model and the teacher model. Following is the step-by-step algorithm to train such an self-ensemble model:

1. First, a student model $M^t_s$ is initialized. This can be any supervised relation extraction model such as CNN, PCNN, Entity Attention (EA) or Bi-GRU Word Attention (BGWA) model.

2. A teacher model $M^t_i$ is cloned from the student model $M^t_s$. We completely detach the weights of the teacher model from the student model.

3. A gradient descent based optimizer is selected to update the parameters of the student model.

4. Loss is calculated based on the cross-entropy loss of the student model for the classification task and a consistency loss between the student model and teacher model.

5. In each training iteration or epoch:

   • In each step or mini-batch:
     - Update the weights of the student model $M^t_s$ using the selected optimizer and the loss function.
     - Update the weights of the teacher model $M^t_i$ as an exponential moving average of the student weights.
   • Evaluate the performance of the teacher model $M^t_i$ on a validation dataset. If we decide to continue the training after evaluation, we use a filtering strategy at this point to remove the noisy samples from the training data. This clean training data is used in the next iteration of the training process.

6. Return the best teacher model $M^t_i$. This teacher model is the self-ensemble version of the student model.

3.2 Loss Function & Updating the Student

We use the negative log-likelihood loss of the relation classification task from the student model ($L_{ce}$) and a mean-squared error based consistency loss between the student and teacher model ($L_{mse}$) to update the student model.

\[
L_{ce} = -\frac{1}{B} \sum_{i=1}^{B} \log(p(r_i|s_i, e^1_i, e^2_i, \theta_s))
\]
\[
L_{mse} = \frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{C} (y^i_{s,j} - y^i_{t,j})^2
\]
\[
L = L_{ce} + L_{mse}
\]
For \( L_{ce}, p(r_i|s_i, e^1_i, e^2_i, \theta_s) \) is the conditional probability of the true relation \( r_i \) when the sentence \( s_i \), two entities \( e^1_i \) and \( e^2_i \), and the model parameters of the student \( \theta_s \) are given. For \( L_{mse}, y^{j,i}_s \) and \( y^{j,i}_t \) are the softmax output of the \( j \) th relation class of \( i \) th training sample in the batch from the student model and the teacher model respectively. \( C \) is number of relation class in the dataset and \( B \) is the batch size. The parameters of the student model \( \theta_s \) are updated based on the combined loss \( L \) using an gradient descent based optimizer. The consistency loss (\( L_{mse} \)) makes sure that output softmax distribution of the student model and teacher model are close to each other, thus maintain the consistency of the output from both models.

3.3 Updating the Teacher

We update the parameters of teacher model \( \theta_t \) based on the exponential moving average of the all previous optimization steps of the student model.

\[
W(\theta^l_t) = \alpha W(\theta^{l-1}_t) + (1 - \alpha) W(\theta^l_s)
\]

where \( W(\theta^l_t) \) and \( W(\theta^l_s) \) are the weights of the parameters of the teacher model and student model respectively after the \( l \) th global optimization step. \( W(\theta^{l-1}_t) \) is the weights of the teacher model parameters up to the \( l - 1 \) th global optimization step. \( \alpha \) is a weight factor to control the contribution of the student model of the current step and the teacher model up to the previous step. At the initial optimization steps of the training, we keep the value of \( \alpha \) low as the self-ensemble model or teacher model is not stable yet and the student model should contribute more. As the training progress and the self-ensemble model becomes stable, we slowly increase the value of \( \alpha \) so that we take the majority contribution from the self-ensemble model itself. We use the following Gaussian curve (He et al., 2018) to ramp up the value of \( \alpha \) from 0 to \( \alpha_{max} \) which is a hyper-parameter of the model.

\[
T = E \times \left\lceil \frac{L}{B} \right\rceil \\
p = 1 - \min(\text{step_idx}, T) \\
\alpha = e^{-5p^2 \alpha_{max}}
\]

Here \( E \) is the epoch count to ramp up the \( \alpha \) from 0 to \( \alpha_{max} \). \( E \) is a hyper-parameter of the model and generally, this is lower than the total number of epochs of the training process. \( L \) is the size of distant supervised training data at the beginning of training, \( B \) is the batch size, and \( \text{step_idx} \) is the current global optimization step count of the training. \( T \) represents the number of global optimization steps required for \( \alpha \) to reach its maximum value \( \alpha_{max} \).

3.4 Noise Filtering Strategy

After each iteration, we use a validation dataset to determine to stop or to continue the training. If we decide to continue the training, then we use the self-ensemble model or the teacher model to filter out noisy samples from the initial training data. This clean training data is used in the next training iteration. We use the self-ensemble model to predict the relation on initial training data for
the filtering process after each iteration. We use the entire initial training data for prediction so that if a training sample is filtered out wrongly in an iteration as a noisy one, it can be used again in subsequent training iterations if the subsequent self-ensemble model predicts the sample as a clean one.

Generally, distantly supervised datasets contain a largely high number of None samples than the valid relation samples. For this reason, we choose a strict filtering strategy for None samples and a lenient filtering strategy for valid relation samples. We consider a None sample as clean if teacher models predict the None relation. Otherwise, this sample is considered as noisy and filtered out from the training set of next iteration. For the valid relations, we consider a sample as clean if the relation assigned by distant supervision belongs to the top $K$ predictions of the teacher model. This clean training data is used in the next training iteration.

4 Student Models

We have used the following state-of-the-art neural relation extraction models as the student model in our filtering framework. These models use three types of embedding vectors: (1) word embedding vector $w \in \mathbb{R}^{d_w}$ (2) a positional embedding vector $u^1 \in \mathbb{R}^{d_u}$ which represents the linear distance of a word from the start token of entity 1 (3) another positional embedding vector $u^2 \in \mathbb{R}^{d_u}$ which represents the linear distance of a word from the start token of entity 2. The sentences are represented using a sequence of vectors $\{x_1, x_2, ..., x_n\}$ where $x_t = w_t || u^1_t || u^2_t$. $\parallel$ represents the concatenation of vectors and $n$ is the sentence length. These token vectors $x_t$ are given as input to all the following models.

4.1 CNN (Zeng et al., 2014)

In this model, convolution operations with max-pooling are applied on the token vectors sequence $\{x_1, x_2, ..., x_n\}$ to obtain the sentence-level feature vector.

\[ c_t = f^T(x_t \parallel x_{t+1} \parallel ... \parallel x_{t+k-1}) \]
\[ c_{\text{max}} = \max(c_1, c_2, ..., c_n) \]
\[ v = [c_{1\text{max}}, c_{2\text{max}}, ..., c_{K\text{max}}] \]

$f$ is a convolutional filter vector of dimension $k(d_w + 2d_u)$ where $k$ is the filter width. The index $i$ moves from 1 to $n$ and produces a set of scalar values $\{c_1, c_2, ..., c_n\}$. The max-pooling operation chooses the maximum $c_{\text{max}}$ from these values as a feature. With $f_k$ number of filters, we get a feature vector $v \in \mathbb{R}^{f_k}$. This feature vector $v$ is passed to feed-forward layer with softmax to classify the relation.

4.2 PCNN (Zeng et al., 2015)

Piecewise Convolutional Neural Network (PCNN) is a modified version of the CNN model described above. Similar to the CNN model, convolutional operations are applied to the input vector sequence. But CNN and PCNN models differ on how the max-pooling operation is performed on the convolutional outputs. Rather than applying a global max-pooling operation on the entire sentence, three max-pooling operations are applied on three segments/pieces of the sentence based on the location of the two entities. This is why this model is called the Piecewise Convolutional Neural Network (PCNN). The first max-pooling operation is applied from the beginning of the sequence to the end of the entity appearing first in the sentence. The second max-pooling operation is applied from the beginning of the entity appearing first in the sentence to the end of the entity appearing second in the sentence. The third max-pooling operation is applied from the beginning of the entity appearing second in the sentence to the end of the sentence. These max-pooled features are concatenated and passed to a feed-forward layer with softmax to determine the relation.
4.3 Entity Attention (EA) (Shen and Huang, 2016)

This model combines the CNN model with an attention network. First, convolutional operations with max-pooling are used to extract the global features of the sentence. Next, attention is applied to the words of the sentence based on the two entities separately. The word embedding of the last token of an entity is concatenated with the embedding of every word. This concatenated representation is passed to a feed-forward layer with tanh activation and then another feed-forward layer with softmax to get a scalar attention score for every word for that entity. The word embeddings are averaged based on the attention scores to get the attentive feature vectors. The CNN-extracted global feature vector and two attentive feature vectors for the two entities are concatenated and passed to a feed-forward layer with softmax to determine the relation.

4.4 Bi-GRU Word Attention (BGWA) (Jat et al., 2017)

This model uses a bidirectional gated recurrent unit (Bi-GRU) (Cho et al., 2014) to capture the long-term dependency among the words in the sentence. The tokens vectors $x_t$ are passed to a Bi-GRU layer. The hidden vectors of the Bi-GRU layer are passed to a bi-linear operator which is a combination of two feed-forward layers with softmax to compute a scalar attention score for each word. The hidden vectors of the Bi-GRU layer are multiplied by their corresponding attention scores for scaling up the hidden vectors. A piecewise convolution neural network (Zeng et al., 2015) is used on top of the scaled hidden vectors to obtain the feature vector. This feature vector is passed to a feed-forward layer with softmax to determine the relation.

5 Experiments

5.1 Datasets

To verify our hypothesis, we need training data that is created using distant supervision, thus noisy and test data which is not noisy, thus human-annotated. If the test data is also noisy, then it will be hard to derive any conclusion from the results. So, we choose the New York Times (NYT) corpus of Hoffmann et al. (2011) for our experiments. This dataset has 24 valid relations and a None relation. The statistics of the dataset is given in Table 2. The training dataset is created by aligning Freebase tuples to NYT articles, but the test dataset is manually annotated. We use 10% of the training data as validation data and the remaining 90% for training.

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>#valid relation instances</td>
<td>100,671</td>
<td>520</td>
</tr>
<tr>
<td>#valid relations</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>#None relation instances</td>
<td>235,172</td>
<td>930</td>
</tr>
</tbody>
</table>

Table 2: The statistics of the NYT dataset.

5.2 Evaluation Metrics

We use precision, recall, and F1 scores to evaluate the performance of models on relation extraction after removing the None labels. We use a confidence threshold to decide if the relation of a test instance belongs to the set of valid relations $R$ or None. If the network predicts None for a test instance, then it is considered as None only. But if the network predicts a relation from the set $R$ and the corresponding softmax score is below the confidence threshold, then the final predicted label is changed to None. This confidence threshold is the one that achieves the highest F1 score on the validation data.

5.3 Parameter Settings

We run word2vec (Mikolov et al., 2013) on the NYT corpus to obtain the initial word embeddings with a dimension of $d_w = 50$ and update the embeddings during training. We set the dimension positional embedding vector at $d_e = 5$. We use $f_k = 230$ convolutional filters of kernel size $k = 3$ for feature extraction whenever we apply the convolution operation. We use dropout in our network with a dropout rate of 0.5, and in convolutional layers, we use the tanh activation function. We train our models with a mini-batch size of 50 and optimize the network parameters using the Adagrad optimizer (Duchi et al., 2011). We want to keep the value of $\alpha_{\text{max}}$ high because when the training progress, we want to increase the contribution of the self-ensemble model compare to the student model. So we set the value of $\alpha_{\text{max}}$ at 0.9. We experiment with $E = \{5, 10\}$ epochs to ramp up the value of $\alpha$ from 0 to $\alpha_{\text{max}}$. We also experiment with $K = \{3, 5\}$ for filtering the valid relation samples during the filtering process after each training iteration. The performance of the self-ensemble model does not vary much with these choices of $E$ or $K$. So we use $E = 5$ and $K = 3$ for final experiments.
Table 3: Precision, Recall, and F1 score comparison of the student models on NYT dataset when trained with self-ensemble filtering framework (SEF column) and when trained independently (Student column). We report the average of five runs with standard deviation. ↑ column shows the absolute % improvement of F1 score over the Student models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Prec.</th>
<th>Rec.</th>
<th>F1</th>
<th>Prec.</th>
<th>Rec.</th>
<th>F1</th>
<th>↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>2.1%</td>
</tr>
<tr>
<td></td>
<td>0.451</td>
<td>0.607</td>
<td>0.518</td>
<td>0.452</td>
<td>0.669</td>
<td>0.539</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.015</td>
<td>0.033</td>
<td>0.021</td>
<td>0.011</td>
<td>0.016</td>
<td>0.005</td>
<td></td>
</tr>
<tr>
<td>PCNN</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>1.1%</td>
</tr>
<tr>
<td></td>
<td>0.431</td>
<td>0.673</td>
<td>0.526</td>
<td>0.432</td>
<td>0.708</td>
<td>0.537</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.013</td>
<td>0.007</td>
<td>0.010</td>
<td>0.009</td>
<td>0.016</td>
<td>0.011</td>
<td></td>
</tr>
<tr>
<td>EA</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>2.1%</td>
</tr>
<tr>
<td></td>
<td>0.437</td>
<td>0.653</td>
<td>0.523</td>
<td>0.444</td>
<td>0.702</td>
<td>0.544</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.012</td>
<td>0.016</td>
<td>0.008</td>
<td>0.008</td>
<td>0.014</td>
<td>0.009</td>
<td></td>
</tr>
<tr>
<td>BGWA</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>2.3%</td>
</tr>
<tr>
<td></td>
<td>0.414</td>
<td>0.680</td>
<td>0.515</td>
<td>0.430</td>
<td>0.720</td>
<td>0.538</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.006</td>
<td>0.021</td>
<td>0.010</td>
<td>0.005</td>
<td>0.014</td>
<td>0.007</td>
<td></td>
</tr>
</tbody>
</table>

5.4 Results

We include the results of our experiments in Table 3. We run the CNN, PCNN, EA, and BGWA models 5 times with different random seeds and report the average with standard deviation in the ‘Student’ column in Table 3. The column ‘SEF’ (Self-Ensemble Filtering) is the average results of 5 runs of CNN, PCNN, EA, and BGWA models with the self-ensemble filtering framework. We see that our SEF framework achieves 2.1%, 1.1%, 2.1%, and 2.3% higher F1 score for the CNN, PCNN, EA, and BGWA models respectively compared to the Student models. If we compare the precision and recall score of the four models, we see that our self-ensemble framework improves the recall score more than the corresponding precision score in each of these four models. These results show the effectiveness of our self-ensemble filtering framework in a distant supervised dataset.

5.5 Self-Ensemble without Filtering

We experiment with how the self-ensemble version of the student models behave without filtering the noisy samples after each iteration. So in this setting, we use the entire distant supervised training data at every iteration. The results are included in Table 4 under the ‘SE’ (Self-Ensemble) column. This result shows that the performance of the four neural models under self-ensemble training without filtering is not much different from the ‘Student’ performance of Table 3. This shows that the filtering of the noisy samples from the training dataset helps to improve the performance of our proposed self-ensemble framework.

Table 4: Precision, Recall, and F1 score of the self-ensemble version of the student models on NYT dataset without noise filtering. We report the average of five runs with standard deviation. ↓ column shows the absolute % decline of F1 score respect to the SEF models (Table 3).

<table>
<thead>
<tr>
<th>Model</th>
<th>Prec.</th>
<th>Rec.</th>
<th>F1</th>
<th>↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>0.448</td>
<td>0.610</td>
<td>0.516</td>
<td>2.3%</td>
</tr>
<tr>
<td></td>
<td>±</td>
<td>±</td>
<td>±</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.012</td>
<td>0.029</td>
<td>0.015</td>
<td></td>
</tr>
<tr>
<td>PCNN</td>
<td>0.432</td>
<td>0.670</td>
<td>0.525</td>
<td>1.2%</td>
</tr>
<tr>
<td></td>
<td>±</td>
<td>±</td>
<td>±</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>0.012</td>
<td>0.005</td>
<td></td>
</tr>
<tr>
<td>EA</td>
<td>0.421</td>
<td>0.647</td>
<td>0.510</td>
<td>3.4%</td>
</tr>
<tr>
<td></td>
<td>±</td>
<td>±</td>
<td>±</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.014</td>
<td>0.017</td>
<td>0.011</td>
<td></td>
</tr>
<tr>
<td>BGWA</td>
<td>0.424</td>
<td>0.689</td>
<td>0.524</td>
<td>1.4%</td>
</tr>
<tr>
<td></td>
<td>±</td>
<td>±</td>
<td>±</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.021</td>
<td>0.020</td>
<td>0.010</td>
<td></td>
</tr>
</tbody>
</table>

5.6 Ensemble vs Self-Ensemble Filtering

Since our SEF framework has an ensemble component, we compare its performance with the ensemble versions of the independent student models. The ‘Ensemble’ column in Table 5 refers to the ensemble results of the 5 runs of each student model. We use the five runs of the models on the test data and average the softmax output of these runs to decide the relation. We see that our SEF framework outperforms the ensemble results for CNN, PCNN, EA, and BGWA with 1.6%, 0.5%, 0.7% and 2.6% F1 score respectively. Here, we should consider the fact that to build an ensemble model, the student models must be run multiple times (5 times in our case). In contrast, self-ensemble models can be built in a single run with little cost of maintaining the moving average of the student model.
<table>
<thead>
<tr>
<th>Model</th>
<th>Prec.</th>
<th>Rec.</th>
<th>F1</th>
<th>↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>0.456</td>
<td>0.613</td>
<td>0.523</td>
<td>1.6%</td>
</tr>
<tr>
<td>PCNN</td>
<td>0.437</td>
<td>0.679</td>
<td>0.532</td>
<td>0.5%</td>
</tr>
<tr>
<td>EA</td>
<td>0.454</td>
<td>0.658</td>
<td>0.537</td>
<td>0.7%</td>
</tr>
<tr>
<td>BGWA</td>
<td>0.410</td>
<td>0.679</td>
<td>0.512</td>
<td>2.6%</td>
</tr>
</tbody>
</table>

Table 5: Precision, Recall, and F1 score of the ensemble version of the student models on NYT dataset. ↓ column shows the absolute % decline of F1 score respect to the SEF models (Table 3).

6 Related Work

There are two approaches for relation extraction (Nayak et al., 2021): (i) Pipeline approaches (Zeng et al., 2014, 2015; Jat et al., 2017; Nayak and Ng, 2019) (ii) Joint extraction approaches (Takanobu et al., 2019; Nayak and Ng, 2020). Most of these models work with distantly supervised noisy datasets. Thus noise mitigation is an important dimension in this area of research. Multi-instance relation extraction is one of the popular methods for noise mitigation. Riedel et al. (2010), Hoffmann et al. (2011), Surdeanu et al. (2012), Lin et al. (2016), Yaghoobzadeh et al. (2017), Vashishth et al. (2018), Wu et al. (2019), and Ye and Ling (2019) used this multi-instance learning concept in their proposed relation extraction models. For each entity pair, they used all the sentences that contain these two entities to find the relation between them. Their goal was to reduce the effect of noisy samples using this multi-instance setting. They used different types of sentence selection mechanisms to give importance to the sentences that contain relation specific keywords and ignore the noisy sentences. But this idea may not be effective if there is only one sentence for an entity pair. Ren et al. (2017) and Yaghoobzadeh et al. (2017) used the multi-task learning approach for mitigating the influence of the noisy samples. They used fine-grained entity typing as an additional task in their model.

Wu et al. (2017) used an adversarial training approach for the same purpose. They add noise to the word embeddings to make the model more robust for distantly supervised training. Qin et al. (2018a) used the generative adversarial network (GAN) to address this issue of the noisy samples in relation extraction. They used a separate binary classifier as a generator in their model for each positive relation class to identify the true positives for that relation and filter out the noisy ones. Qin et al. (2018b) used reinforcement learning to identify the noisy samples for the positive relations and then use the identified noisy samples as unlabelled data in their model. Shang et al. (2020) used a clustering approach to identify the noisy samples. They assign the correct relation label to these noisy samples and use them as additional training data in their model. Different from these approaches, we propose a student-teacher framework that can work with any supervised neural network models to address the issue of noisy samples in distantly-supervised datasets.

7 Conclusion

In this work, we propose a self-ensemble based noisy samples filtering framework for distantly supervised relation extraction. Our framework identifies the noisy samples during training and removes them from the training data in the following iterations. This framework can be used with any supervised relation extraction models. We run experiments using several state-of-the-art neural models with this proposed filtering framework on the distantly supervised New York Times dataset. The results show that our proposed framework improves the robustness of these models and increases their F1 score on the relation extraction task.

Acknowledgments

This research is supported by A*STAR under its RIE 2020 Advanced Manufacturing and Engineering (AME) programmatic grant RGAST2003, (award # A19E2b0098, project K-EMERGE: Knowledge Extraction, Modelling, and Explainable Reasoning for General Expertise).

References

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In IJCAI.


Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL and IJCNLP*.


Tapas Nayak and Hwee Tou Ng. 2019. Effective attention modeling for neural relation extraction. In *CoNLL*.

Tapas Nayak and Hwee Tou Ng. 2020. Effective modeling of encoder-decoder architecture for joint entity and relation extraction. In *AAAI*.


Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, Tarek F. Abdelzaher, and Jiawei Han. 2017. CoType: Joint extraction of typed entities and relations with knowledge bases. In *WWW*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *ECML and KDD*.


Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*.


Abstract

Biomedical Named Entities are complex, so approximate matching has been used to improve entity coverage. However, the usual approximate matching approach fetches only one matching result, which is often noisy. In this work, we propose a method for biomedical NER that fetches multiple approximate matches for a given phrase to leverage their variations to estimate entity-likeness. The model uses pooling to discard the unnecessary information from the noisy matching results, and learn the entity-likeness of the phrase with multiple approximate matches. Experimental results on three benchmark datasets from the biomedical domain, BC2GM, NCBI-disease, and BC4CHEMD, demonstrate the effectiveness. Our model improves the average F-measures by up to 0.21 percentage points compared to a BioBERT-based NER.

1 Introduction

In the biomedical field, obtaining labelled data is very costly. Biomedical Named Entities (NEs) are complex and new NEs are continuously increasing in significant numbers, leading to unknown-word issues in Biomedical Named Entity Recognition (BioNER) tasks. One reason why biomedical NEs are complex is that they have many variations with the interchangeability of Roman numbers and Latin characters, spaces and hyphens, etc. The number of new biomedical research papers is increasing, wherein approximately two papers per minute, resulting in more than 1 million papers each year, are added to the PubMed database (Landhuis, 2016). With this number of publications, new NEs are constantly being reported.

In the last few years, NER using pre-trained language models (LMs), such as BERT (Devlin et al., 2018), ELMo (Peters et al., 2018), and Flair (Akbik et al., 2019), has shown state-of-the-art performance. In the biomedical domain, pre-trained LMs such as BioBERT (Lee et al., 2019a) and BioELMo (Jin et al., 2019), which are BERT and ELMo trained on a biomedical domain text, have achieved the state-of-the-art performance in many biomedical natural language processing tasks including NER. However, only using previously trained LMs cannot cover the continuously increasing new entities due to complex characteristics of biomedical NEs, lead to unknown words problem. Despite being used as approaches to avoid unknown words problem, subword segmentation (Sennrich et al., 2015; Kudo and Richardson, 2018) methods consider subwords represented as unique IDs, but not words or their synonyms. therefore, it is difficult for subword or character based LMs to cover biomedical NEs, which are complex and contain various of expression described in section 3. Moreover, LM pre-training is costly, time-consuming, and computationally expensive. Training BioBERT on biomedical corpora based on the BERT model requires 10 to 23 days on eight NVIDIA V100 GPUs (Lee et al., 2019a).

To deal with the complex and continuously increasing entities, the use of dictionary-based approaches can be an effective approach in previous works (Collobert et al., 2011; Rijhwani et al., 2020). In contrast to pre-training models, we can cover new NEs by adding entries to the dictionary, without needing time-consuming pre-training. There are two types of dictionary application methods: exact matching and approximate matching. Exact matching has been incorporated into neural NER (Collobert et al., 2011; Chiu and Nichols, 2016; Wu et al., 2018) and non-neural NER methods (Uchimoto et al., 2000) to improve accuracy.

Exact matching cannot totally cover all of the complex and newly-created NEs. In the biomedical domain, new NEs are created by modifying the endings of the existing one. For example, the new gene TAAR7P was named by modifying the
ending of the existing gene TAAR8. To improve the coverage of entities, approximate matching has been used to manage new NEs in non-neural NER (Cohen and Sarawagi, 2004). However, the approximate matching approach fetches only one matching result, which cannot cover all variations of NEs. For example, NEs “Type-1 angiotensin II receptor-associated protein” have many variations such as “Type I angiotensin II receptor associated protein”, “Type-1 angiotensin 2 receptor associated protein”, and “Type 1 angiotensin II receptor-associated protein”. Also, approximate matching results are often noisy.

In this paper, we propose a method to improve neural BioNER by learning the entity-likeness of a given input sentence using multiple approximate matches of the input sentence with a dictionary. We define the entity-likeness as the degree to which a certain input sentence is likely to appear in the dictionary. It is estimated from matching results between the input sentence and entities in the dictionary.

We evaluated our method with three biomedical domain benchmarks, i.e., BC2GM, NCBI-disease, and BC4CHEMD dataset. The experimental results show the effectiveness of our approach. It improves F-measures by up to +0.21 points on the biomedical benchmark, and +2.2 points when probing the biomedical ELMo (Jin et al., 2019), which is a recent state-of-the-art pre-training method.

2 Related Work

For the NER task, previous studies have examined the application of dictionaries in machine learning. Dictionary matching was employed in SVM-based NER (Ratinov and Roth, 2009) and partial matching computed by distance feature between a token and entity in dictionary was considered in semi-Markov extraction processes (Cohen and Sarawagi, 2004).

Dictionary matching is also used in Neural NER approaches. Liu et al. added a pre-trained module that softly matches the gazetteers to the semi-Markov CRF-based segmental NER task. Soft matching of gazetteers is also used in the work of Rijhwani et al. (2020) for low-resource NER. Exact matching was used by Collobert et al. (2011) they use a network layer to map words of dictionary into feature vectors by a lookup table operation and train the features as input in their model. Chiu and Nichols proposed the use of the longest matching, including partial lexicon matching in neural networks. Each word vector has dimensions to express dictionary matching.

In the CRF-based sequence labeling model for NER, the clustering results of phrases in the search engine query logs were used as features by Lin and Wu (2009). To improve word representation, a word embedding learning method that leverages information from relevant lexicons to phrase embedding was proposed by Passos et al. (2014). Hand-crafting features obtained from gazetteers were also incorporated to model additional information in the named entity (Wu et al., 2018; Shang et al., 2018).

Related to approaches employing approximate string matching in Biomedical NER, Tsuruoka and Tsujii proposed a method to recognize entity candidates by approximate searching and filtering out false positives using a binary classifier. Yang et al. used approximate string matching and added pre- and post-keywords for each bio-entity name to expand the coverage of the dictionary. Xu et al. constructed a dictionary attention layer to incorporate exact dictionary matching and a document-level attention mechanism to improve disease NER.

Approaches based on neural network were also applied for Biomedical NER (Habibi et al., 2017; Crichton et al., 2017; Wang et al., 2018). For a transformer-based approach, Khan et al. used a shared transformer encoder to capture the embedding vector of each token in input sentence and task specific linear layers to generate representations of multi-tasks including Biomedical NER.

Differing from these works, we propose a method to learn the entity-likeness of a sentence by leveraging multiple approximate matches of the sentence with one or multiple dictionaries. Recent approaches based on pre-training for specific domains, such as biomedical (Lee et al., 2019a; Jin et al., 2019), clinical (Huang et al., 2019) and scientific (Beltagy et al., 2019), have shown high levels of accuracy; our method is complementary to these approaches.

3 NEs in Biomedical Domain

Biomedical NEs are complex and ambiguous due to the following characteristics:

**Variation of Expression** Biomedical NEs have various synonyms, including abbreviations, interchangeability of Roman numbers and Latin characters, insertions and deletions of hyphens and spaces, and changes in word order. For example, the gene
“Angiotensin II Receptor Type 1” has the official name “AGTR1”, as well as more than ten other names, e.g., AGTR-1, Type-1 Angiotensin II Receptor, Angiotensin Receptor 1B, and AT1 Receptor. Even if the dictionary is further expanded, exact matching cannot entirely cover all possible variations of NEs.

**Composite Mentions** NEs in the biomedical domain are frequently connected by “and,” “or” in a single span which refers to more than one entity. For example, “alpha and beta globin” refers to “alpha globin” and “beta globin”.

**Nested NEs** Nested NEs (Kim et al., 2003; Ringland et al., 2019), where one NE is completely contained by the other, are also commonly used in biomedical data. For example, both “adenylate cyclase activating polypeptide 1” and “adenylate cyclase” are the names of proteins.

**Entity ambiguity** The same mention may often refer to many different entities depending on context. For example, “VHL” can be either a disease name “Von Hippel–Lindau (VHL) disease” or a gene name “VHL gene” depending on context.

NEs in the biomedical domain are continuously increasing in number every year. When using exact matching or pre-trained LMs for BioNER, it is difficult to sufficiently cover all possible combinations of NEs, leading to the omission of NE recognition.

4 Learning Entity-likeness with Multiple Approximate Matches

The concept of our approach is that the entity-likeness of a given input sentence can be estimated by its maximal similarity to entities in a dictionary. Our motivation is to assign the entity-likeness to each word of the input sentence.

The overall flow of the proposed approach is as follows:

1. Given an input sentence, we first fetch matching results between the input sentence and a specified dictionary.

2. We create matching patterns based on the matching results, and assign them to each word in the input sentence. The matching pattern is a label that indicates how each word matches with the dictionary.

3. For each word in the input sentence, we build a vector for predicting entity-likeness from the multiple matching patterns by a pooling operation.

4. We build an NER model learning both vector of entity-likeness and contextual embedding derived from pre-trained LMs.

4.1 Creating Multiple Approximate Matches

Given an input sentence, we first fetch the matching results between the input sentence and entities in a dictionary. Since we cannot specify which part of the input sentence contains the entity, we calculate the string similarity of all continuous word level N-grams ($N \leq 5$) in the input sentence with all dictionary entries. The matching returns entries whose similarity with the $N$-gram is larger than a specified threshold $\tau$. We regard a match of $N$-gram with an entity with threshold 1.0 as an exact matching.

By employing the multiple approximate matchings of $N$-gram with the dictionary, it is possible to obtain useful information about the multiple matches for estimating the entity-likeness of the $N$-grams, especially in the case of predicting a new NE which is similar to the existing one. For example, we can obtain information on the interchange-ability of Greek or Roman characters in NEs from dictionary entries “beta-1 Adrenergic Receptor”, “β-1 Adrenergic Receptor” and other synonyms. The information is useful for recognizing the unknown NE “α-1 Adrenergic Receptor”.

4.2 Creating Dictionary Matching Patterns

Based on the matching results of $N$-grams ($N \leq 5$) with a dictionary obtained in section 4.1, we create a set of dictionary matching patterns that includes the information of the dictionary that is used, the types of matching, and the matching position; this information is assigned to each word in the input sentence. The type of matching is set to “Exact” if the $N$-gram exactly matches the dictionary entry, otherwise it is set to “Approximate”. There are three types of matching positions (B (Beginning), I (Inside), and E (Ending)) which indicate the position of the word in the $N$-gram.

For example, as shown in Figure 1, the input sentence “EGFR is epidermal growth factor receptor” is matched with a gene/protein dictionary. The gene/protein dictionary includes entries such as “epidermal growth factor receptor substrate 1”.

Note that an $N$-gram can be matched with one or multiple dictionaries when we have two or more dictionaries.
15,” “epidermal growth factor receptor GRB-7,” etc. As shown in Figure 1, 3-gram \textbf{N1} with the beginning word \(w_0\) “epidermal” exactly matches with gene/protein dictionary entry \	extbf{M1} and it approximately matches with entries \	extbf{M3}, \	extbf{M4} and \textbf{M5}. The matching result of the 3-gram \textbf{N1} assigns matching patterns: “Gene-Exact-B” and “Gene-Approximate-B” to \(w_0\).

In the same way, 4-gram \textbf{N2} approximately matches with dictionary entries \textbf{M6} and \textbf{M7}. In this case, the word “epidermal” is inside the \textbf{N2} and therefore the matching result of the \textbf{N2} assigns matching patterns: “Gene-Approximate-I” to \(w_0\).

Based on matching results between all \(N\)-grams of the input sentence and the dictionary, we can obtain a set of matching patterns for each word in the input sentence. The possible matching patterns for each word are \{Number of dictionaries\} \times \{Exact, Approximate\} \times \{B, I, E\}. For example, in Figure 1, a set of matching patterns with the Gene dictionary for the third word “epidermal” are \{“Gene-Exact-B,” “Gene-Approximate-B,” “Gene-Approximate-I”\}.

4.3 Representation of Multiple Matching Patterns

After creating sets of dictionary matching patterns corresponding to each word, we build a representation for the dictionary matching patterns.

Suppose each word \(w_i\) corresponds to a subset of matching patterns \(S_i \subset S\), where \(S\) is the possible matching patterns, \(S_i\) is obtained in section 4.2. Here, \(S_i\) represents the likeliness of that the word forms a part of entities. \(E_i\) corresponds to embeddings of \(S_i\):

\[
E_i = \{\text{emb}(s) | s \in S_i\} \tag{1}
\]

where \(\text{emb}(\cdot)\) indicates an embedding operation. In experiments, embedding \(\text{emb}(s)\) is randomly initialized from a normal distribution but not fine-tuned.

Next, we build a vector representation \(D_i\) of entity-likeness by pooling the embeddings \(E_i\); \(D_i\) has the same dimension as \(E_i\):

\[
D_i = f_{pool}(E_i) \tag{2}
\]

where \(f_{pool}\) is a pooling operation.

The aim of the pooling is to aggregate information for learning from various matching patterns. In order to investigate the effect of various pooling functions, we consider four types of pooling: Sum, Max, Average and Convolution.

**Sum Pooling** It is expected that summarizing all features of the possible matching pattern embeddings gives information for estimating entity-likeness of words.

\[
f_{sum}(E_i) = \sum_{v \in E_i} v \tag{3}
\]
Figure 2: Illustration of the proposed model architecture. $T_i$ and $D_i$ are the corresponding contextual word embedding module and dictionary matching pattern module for each word $w_i$ in the input sentence, respectively. $V_i$ represents interaction between each word and its entity-likeness. The model predicts the token-level NE label, $l_i$. $am_1 B$, $am_1 I, ...$ are embeddings of matching patterns.

Max Pooling Instead of sum pooling, we use max pooling to compose the set of matching pattern embeddings:

$$f_{\text{max}}(E_i) = \max(E_i) \quad (4)$$

Average Pooling In the same way, we consider the average variation of the pooling method:

$$f_{\text{avg}}(E_i) = \text{avg}(E_i) \quad (5)$$

Convolution As a way to combine embeddings, we apply 1-D convolution over the set of matching pattern embeddings to build the dictionary matching embedding:

$$f_{\text{conv}}(E_i) = \text{Conv1d}(E_i) \quad (6)$$

4.4 Learning Representations of Entity-likeness with NER

Figure 2 shows the overview of our method. Given the output of the contextual word embedding $T_i$, and vector representation of entity-likeness $D_i$, the label prediction module predicts the IOB2 labels of input sentence $w_i$. By learning $T_i$ and $D_i$ together, it is possible to recognize new NEs which were not in the dictionary or training data of LMs. For the pre-trained LMs, we use BioBERT (Lee et al., 2019a) or BioELMo (Jin et al., 2019) depending on experiments.

The layer numbers and the internal details of the label prediction layer vary depending on the used pre-trained LMs. We follow the settings of the original studies (Lee et al., 2019a; Jin et al., 2019). In the case of BioBERT, we use a single linear layer to compute token level IOB2 probabilities. In the case of BioELMo, we follow the probing settings in the work of Jin et al. (2019). We use several linear layers to compute the probabilities.

5 Experiments

In this section, we conduct three experiments. Experiment 1 confirms the effectiveness of learning both entity-likeness and contextual embedding for BioNER. Also, we want to confirm if applying appropriate pooling operations can reduce noise in the case of approximate matching. Experiment 2 confirms portability by using our method with different pre-trained LMs. Experiment 3 confirms the effectiveness of our method not only in the biomedical domain but also in the general domain.

For pre-trained LMs, we employed BioBERT and BioELMo trained on PubMed and PMC biomedical articles. For experiments on a general domain dataset, we applied the pre-trained BERT base cased LMs.

5.1 Datasets

In this study, the results were obtained by adopting the proposed and BioBERT-based methods to three benchmark biomedical datasets, BC2GM, NCBI-disease, and BC4CHEMD, which are exclusively annotated with protein, disease, and chemical entities, respectively. For the general domain, we used the CoNLL 2003 dataset (Tjong Kim Sang and De Meulder, 2003). Table 1 shows the size of the datasets. All datasets are publicly available.

5.2 Dictionary

We consider the dictionary as a set of names including synonyms of the entities, e.g., Gene, Disease, and Drug. In the biomedical domain, there are several publicly available databases that can be used to create dictionaries. The dictionaries are built from

2https://github.com/cambridgeltl/MTL-Bioinformatics-2016
Table 1: Size of NER datasets used in the experiments. The numbers are in sentences.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>train</th>
<th>dev</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC2GM</td>
<td>12,574</td>
<td>2,519</td>
<td>5,038</td>
</tr>
<tr>
<td>NCBI-disease</td>
<td>5,424</td>
<td>923</td>
<td>940</td>
</tr>
<tr>
<td>BC4CHEMD</td>
<td>30,682</td>
<td>30,639</td>
<td>26,364</td>
</tr>
<tr>
<td>CoNLL 2003</td>
<td>14,987</td>
<td>3,466</td>
<td>3,684</td>
</tr>
</tbody>
</table>

the databases. Therefore, we do not need to create and maintain dictionaries from scratch.

We construct dictionaries for genes/proteins, diseases, and drugs, to train the proposed model on the BC2GM, NCBI-disease, and BC4CHEMD datasets, respectively. Further, three dictionaries of person (PER), location (LOC), and organization (ORG) are built to train the proposed model on the CoNLL 2003 dataset.

**Gene/protein dictionary** We created a gene/protein dictionary from public databases: Human Gene Nomenclature (HGNC) and NCBI Entrez Gene (Maglott et al., 2019). HGNC is a database containing unique names and alias names for human genes. NCBI Entrez Gene is the National Center for Biotechnology Information (NCBI)’s database for gene-specific information (Maglott et al., 2011). We extracted gene names, their symbols, alias symbols, and alias names to build our gene/protein dictionary. The dictionary contains 292,853 gene entity surfaces.

**Disease dictionary** We built a disease dictionary based on Human Disease Ontology (LM et al., 2019). Our disease dictionary is built from disease names and their synonyms based on the ontology with 30,426 disease entities.

**Drug dictionary** For the drug dictionary, we used DrugBank Vocabulary ³ from DrugBank (DS et al., 2019). We entered common names and synonyms as drug names into the dictionary. The dictionary contains 26,235 drug entities.

**PER, LOC, and ORG dictionaries** We constructed three dictionaries on person (PER), location (LOC), and organization (ORG) from the DBpedia database ⁴ to train the proposed model on the CoNLL 2003 dataset. We used categories from the 2019-8-30 Version and extracted categories that include keywords such as “Person,” “Organization,” and “Places” to construct the dictionaries. The dictionary consists of 710,492 PER, 37,687 ORG, and 69,028 LOC entities.

### 5.3 Experimental Setting

To obtain multiple approximate matches of the input sentence and dictionary, we used Simstring (Okazaki and Tsujii, 2010), an approximate string matching library that searches for similarities between a set of characters (e.g., “cosine,” “jaccard”) with a query string length exceeding a specified threshold. Simstring is known as a fast and efficient algorithm for approximate dictionary matching.

We used Simstring to obtain matching results for N-gram (N ≤ 5) with the dictionary. The cosine similarity threshold between N-grams of the input sentence and dictionary entries was empirically set to 0.8. This is because the threshold value of 0.8 revealed good results during preliminary experiments. Next, we created a set of matching patterns based on the matching results.

For hyperparameter tuning, entity-likeness representation dimension sizes of 50, 100, and 300, and batch sizes of 16 and 32, were selected. Therein, we decided the parameter for entity-likeness representation and batch size are 100 and 32, respectively. Contextual word embedding derived from the pre-trained model is concatenated with 100-dimensional entity-likeness representation embeddings, and then fed into a label prediction layer. We trained for 20 epochs and the NER results were averaged over five seeds.

All experiments were conducted using a single NVIDIA GeForce RTX 16 GB GPU. Pytorch version was 1.4.0. We used the HuggingFace PyTorch implementation of (Wolf et al., 2019) ⁵ to conduct the experiments.

**Experiment 1: Learning Entity-likeness with BioBERT** We followed the recipe of Lee et al. (2019a) to train the model with the following hyperparameters: learning rates of 1e-5; batch sizes of 32; and weight-decay of 0.001. We used the pre-trained model BioBERT v1.0 (Wiki + Books + PubMed 200K + PMC 270K) ⁶ as a contextual word embedding.

³https://www.drugbank.ca/releases/5-1-4/downloads/all-drugbank-vocabulary
⁴https://downloads.dbpedia.org/repo/lts/generic/
⁵https://github.com/huggingface/transformers
⁶https://github.com/naver/biobert-pretrained
For the approach using the approximate matching result, we compared our method with Liu et al. (2019). They proposed a pre-training sub-tagger softdict that softly matches a sentence with gazetteers for NER. This sub-tagger plays the role of an approximate dictionary look-up. Softdict is trained on gazetteers and non-entity N-grams sampled from the corpus.

They sampled 1 million non-entity N-grams from 14,987 sentences in the CoNLL 2003 training data. For each dataset, we sampled non-entity N-grams using the same ratio of data size and sample size. Following the settings in their work, we used pre-trained 50-dimensional Glove word embedding (Pennington et al., 2014), contextualized ELMo embedding, a convolutional character encoder and the pre-trained softdict to train the NER model.

Experiment 2: Learning Entity-likeness with BioELMo and Bio_word2vec We confirmed the performance of the proposed method with other pre-trained LMs. We conducted experiments using contextual embeddings from pre-trained models BioELMo (Jin et al., 2019) \(^7\) and Bio_word2vec (Pyysalo et al., 2013) \(^8\). We kept the default hyperparameters settings in Jin et al.’s work, with a batch size of 32, Adam learning rate of 0.002, and training for 10 epochs. The embedding derived from BioELMo or Bio_word2vec is concatenated with 100-dimensional entity-likeness representation embeddings and then are fed to four feed-forward layers and a CRF output layer.

Experiment 3: Learning Entity-likeness with BERT For experiments on the CoNLL 2003 dataset, a pre-trained BERT-based case was used instead of BioBERT. Hyperparameters were set the same as for learning entity-likeness with Experiment 1.

5.4 Results

For learning entity-likeness with BioBERT, we evaluated the accuracy of the results with an entity-level F-measures. For learning entity-likeness with BioELMo and Bio_word2vec, we used the official evaluation codes of BC2GM, which contain multiple ground-truth tags to calculate F-measures, following the work of Jin et al. (2019).

The experimental results are presented in Tables 2, 3 and 4. In Table 2, the F-measures were obtained in the experiments conducted based on the Pytorch implementation library of (Wolf et al., 2019); the best scores are denoted in bold. The scores are almost the same with scores reported in (Lee et al., 2019b), which are not the scores reported in the original BioBERT papers (Lee et al., 2019a).

The difference in scores of the original paper (Lee et al., 2019a) and (Lee et al., 2019b) is due to the neural network implementation library (Pytorch-based or TensorFlow-based), the implementation framework (HuggingFace, etc.), and the GPU architecture and setting of the random seed.

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>BioBERT</td>
<td>82.34 ±0.02</td>
<td>84.82 ±0.02</td>
<td>83.56 ±0.02</td>
</tr>
<tr>
<td>Liu et al.</td>
<td>79.63 ±0.002</td>
<td>81.09 ±0.009</td>
<td>80.35 ±0.004</td>
</tr>
<tr>
<td>Exa-Sum</td>
<td>82.58 ±0.05</td>
<td>84.65 ±0.05</td>
<td>83.60 ±0.02</td>
</tr>
<tr>
<td>Exa-Max</td>
<td>82.54 ±0.01</td>
<td>84.61 ±0.02</td>
<td>83.56 ±0.00</td>
</tr>
<tr>
<td>Exa-Avg</td>
<td>82.52 ±0.01</td>
<td>84.61 ±0.02</td>
<td>83.55 ±0.00</td>
</tr>
<tr>
<td>Exa-Conv</td>
<td>82.56 ±0.03</td>
<td>84.61 ±0.06</td>
<td>83.57 ±0.04</td>
</tr>
<tr>
<td>App-Sum</td>
<td>82.69 ±0.01</td>
<td>84.71 ±0.01</td>
<td>83.69 ±0.02</td>
</tr>
<tr>
<td>App-Max</td>
<td>82.57 ±0.01</td>
<td>84.66 ±0.03</td>
<td>83.65 ±0.00</td>
</tr>
<tr>
<td>App-Avg</td>
<td>82.51 ±0.04</td>
<td>84.60 ±0.02</td>
<td>83.58 ±0.00</td>
</tr>
<tr>
<td>App-Conv</td>
<td>82.54 ±0.04</td>
<td>84.66 ±0.01</td>
<td>83.58 ±0.02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>BioBERT</td>
<td>86.67 ±0.06</td>
<td>90.28 ±0.02</td>
<td>88.44 ±0.03</td>
</tr>
<tr>
<td>Liu et al.</td>
<td>85.21 ±0.006</td>
<td>87.01 ±0.005</td>
<td>86.10 ±0.003</td>
</tr>
<tr>
<td>Exa-Sum</td>
<td>86.40 ±0.02</td>
<td>90.37 ±0.02</td>
<td>88.34 ±0.02</td>
</tr>
<tr>
<td>Exa-Max</td>
<td>86.67 ±0.06</td>
<td>90.30 ±0.06</td>
<td>88.44 ±0.02</td>
</tr>
<tr>
<td>Exa-Avg</td>
<td>86.68 ±0.04</td>
<td>90.38 ±0.05</td>
<td>88.49 ±0.10</td>
</tr>
<tr>
<td>Exa-Conv</td>
<td>86.57 ±0.05</td>
<td>90.26 ±0.07</td>
<td>88.38 ±0.06</td>
</tr>
<tr>
<td>App-Sum</td>
<td>86.74 ±0.06</td>
<td>90.64 ±0.06</td>
<td>88.65 ±0.05</td>
</tr>
<tr>
<td>App-Max</td>
<td>86.39 ±0.02</td>
<td>90.58 ±0.02</td>
<td>88.43 ±0.03</td>
</tr>
<tr>
<td>App-Avg</td>
<td>86.73 ±0.04</td>
<td>90.51 ±0.05</td>
<td>88.58 ±0.01</td>
</tr>
<tr>
<td>App-Conv</td>
<td>86.46 ±0.06</td>
<td>90.51 ±0.12</td>
<td>88.49 ±0.08</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>BioBERT</td>
<td>91.89 ±0.06</td>
<td>90.95 ±0.04</td>
<td>91.41 ±0.02</td>
</tr>
<tr>
<td>Liu et al.</td>
<td>88.78 ±0.06</td>
<td>89.02 ±0.02</td>
<td>88.89 ±0.03</td>
</tr>
<tr>
<td>Exa-Sum</td>
<td>91.79 ±0.10</td>
<td>91.08 ±0.05</td>
<td>91.43 ±0.03</td>
</tr>
<tr>
<td>Exa-Max</td>
<td>91.92 ±0.06</td>
<td>90.93 ±0.10</td>
<td>91.43 ±0.05</td>
</tr>
<tr>
<td>Exa-Avg</td>
<td>91.90 ±0.08</td>
<td>91.00 ±0.10</td>
<td>91.44 ±0.00</td>
</tr>
<tr>
<td>Exa-Conv</td>
<td>91.86 ±0.06</td>
<td>90.94 ±0.03</td>
<td>91.45 ±0.00</td>
</tr>
<tr>
<td>App-Sum</td>
<td>91.81 ±0.10</td>
<td>91.11 ±0.05</td>
<td>91.45 ±0.02</td>
</tr>
<tr>
<td>App-Max</td>
<td>91.94 ±0.06</td>
<td>91.01 ±0.08</td>
<td>91.47 ±0.01</td>
</tr>
<tr>
<td>App-Avg</td>
<td>91.88 ±0.10</td>
<td>91.06 ±0.10</td>
<td>91.47 ±0.00</td>
</tr>
<tr>
<td>App-Conv</td>
<td>91.85 ±0.10</td>
<td>91.03 ±0.08</td>
<td>91.44 ±0.00</td>
</tr>
</tbody>
</table>

Table 2: Experimental results of the proposed method with BioBERT-base model on three biomedical datasets BC2GM, NCBI-disease, and BC4CHEMD. Cells represent Precision, Recall and F-measure with standard deviation on each test set, respectively. Exa and App denote Exact and Approximate, respectively.

---

\(^7\)https://github.com/Andy-jqa/bioelmo  
\(^8\)http://bio.nlplab.org
Table 3: Results of learning entity-likeness by probing BioELMo and Bio_word2vec on the BC2GM dataset. Cells represent Precision, Recall and F-measure with standard deviation. Exa and App denote Exact and Approximate, respectively.

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BioELMo</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exa-Sum</td>
<td>89.6 ±0.61</td>
<td>90.4 ±0.56</td>
<td>90.0 ±0.06</td>
</tr>
<tr>
<td>Exa-Max</td>
<td>90.4 ±0.18</td>
<td>89.9 ±0.33</td>
<td>90.1 ±0.24</td>
</tr>
<tr>
<td>Exa-Avg</td>
<td>90.6 ±0.38</td>
<td>89.7 ±0.53</td>
<td>90.2 ±0.35</td>
</tr>
<tr>
<td>Exa-Con</td>
<td>89.5 ±0.72</td>
<td>89.9 ±0.78</td>
<td>89.7 ±0.02</td>
</tr>
<tr>
<td>App-Sum</td>
<td>91.1 ±0.68</td>
<td>90.0 ±0.67</td>
<td><strong>90.6 ±0.25</strong></td>
</tr>
<tr>
<td>App-Max</td>
<td>90.0 ±0.34</td>
<td><strong>90.4 ±0.50</strong></td>
<td>90.2 ±0.29</td>
</tr>
<tr>
<td>App-Avg</td>
<td>89.3 ±0.55</td>
<td>90.3 ±0.18</td>
<td>89.8 ±0.19</td>
</tr>
<tr>
<td>App-Con</td>
<td>89.1 ±0.10</td>
<td>90.0 ±0.30</td>
<td>89.5 ±0.10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bio_word2vec</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exa-Sum</td>
<td>86.3 ±0.30</td>
<td>80.3 ±0.24</td>
<td>83.2 ±0.16</td>
</tr>
<tr>
<td>Exa-Max</td>
<td>86.2 ±0.28</td>
<td>79.9 ±0.11</td>
<td>82.9 ±0.10</td>
</tr>
<tr>
<td>Exa-Avg</td>
<td>86.2 ±0.06</td>
<td>80.2 ±0.30</td>
<td>83.1 ±0.07</td>
</tr>
<tr>
<td>Exa-Con</td>
<td>84.9 ±0.36</td>
<td>80.7 ±0.38</td>
<td>82.7 ±0.06</td>
</tr>
<tr>
<td>App-Sum</td>
<td>85.7 ±0.10</td>
<td><strong>81.3 ±0.33</strong></td>
<td><strong>83.4 ±0.09</strong></td>
</tr>
<tr>
<td>App-Max</td>
<td>85.4 ±0.51</td>
<td>80.7 ±0.13</td>
<td>83.0 ±0.11</td>
</tr>
<tr>
<td>App-Avg</td>
<td>85.9 ±0.28</td>
<td>80.9 ±0.51</td>
<td>83.3 ±0.13</td>
</tr>
<tr>
<td>App-Con</td>
<td>85.2 ±0.29</td>
<td>81.2 ±0.07</td>
<td>83.2 ±0.10</td>
</tr>
</tbody>
</table>

Table 3: Results of learning entity-likeness by probing BioELMo and Bio_word2vec on the BC2GM dataset. Cells represent Precision, Recall and F-measure with standard deviation. Exa and App denote Exact and Approximate, respectively.

As listed in Tables 2, 3 and 4, learning both exact matching and approximate matching outperforms BioBERT-based methods and improves F-measures by up to +0.13, +0.21 and +0.06 points on the three biomedical benchmarks BC2GM, NCBI-disease and BC4CHEMD, respectively; BioELMo and Bio_word2vec improve F-measures by up to +2.2 and +4.9 points on BC2GM; BERT-based methods improve F-measures by up to +0.25 points on CoNLL 2003.

6 Discussion

The experimental results indicate that, in the case of exact matching, F-measures are not highly different for the four types of pooling. As shown in Table 2, 3 and 4, sum pooling obtains the best results in the case of approximate matching. It is considered to be more informative for summarizing all features of the possible approximate matching patterns to estimate entity-likeness. Precision is improved in exact matching while recall is improved in approximate matching. In approximate matching, even though the matching results are noisy, tuning to select the appropriate pooling can help minimize noise. Our approach has effectiveness for small datasets such as NCBI-disease, and multi-category datasets such as CoNLL 2003, where F-measures improved by up to +0.21 and +0.25 points, respectively.

In Table 2, the improvement of F-measures is not significant on the BC4CHEMD dataset. It is thought that this is because approximate matching of N-gram (N ≤ 5) returns only dictionary entries which approximately match with N-gram only up to 5-words, while there are drug names whose length are much longer than 5-gram in BC4CHEMD dataset. For datasets containing long NEs, it is necessary to set N-grams with larger values.

Our approach has effectiveness for small datasets with complicated NEs. In reality, obtaining large-scale domain specific data like BC2GM and BC4CHEMD is very costly, while NEs in the biomedical domain are complex and continuously increasing every year.

7 Conclusion

In this paper, we proposed a new approach: learning the entity-likeness of phrases in sentences by using multiple approximate matching results. The experiments show three properties. The approach has portability with various pre-trained LMs. Our Sum pooling methods efficiently filter noisy approximate matching results for learning entity-likeness. Our approach effectively works particularly on small datasets, not only in the biomedical area but also in more general domains. Moreover, our approach does not require expensive computation. We hope that the proposed approach can contribute to identifying NEs in such cases.
References


HUGO Gene Nomenclature Committee (HGNc), European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Wellcome Genome Campus, Hinxton, Cambridge CB10 1SD, and United Kingdom www.genenames.org. 2019. Hgnc gene name database.


Jingbo Shang, Liyuan Liu, Xiang Ren, Xiaotao Gu, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. CoRR, abs/1809.03599.


Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei Han. 2018. Cross-type biomedical named entity recognition with deep multi-task learning. CoRR, abs/1801.09851.


Extending a Text-to-Pictograph System to French and to Arasaac

Magali Norré
CENTAL, UCLouvain, Belgium
FTI, UNIGE, Switzerland

Vincent Vandeghinste
Instituut voor de Nederlandse Taal, The Netherlands

Pierrette Bouillon
FTI, UNIGE, Switzerland

Thomas François
CENTAL, UCLouvain, Belgium

Abstract

We present an adaptation of the Text-to-Picto system, initially designed for Dutch, and extended to English and Spanish. The original system, aimed at people with an intellectual disability, automatically translates text into pictographs (Sclera and Beta). We extend it to French and add a large set of Arasaac pictographs linked to WordNet 3.1. To carry out this adaptation, we automatically link the pictographs and their metadata to synsets of two French WordNets and leverage this information to translate words into pictographs. We automatically and manually evaluate our system with different corpora corresponding to different use cases, including one for medical communication between doctors and patients. The system is also compared to similar systems in other languages.

1 Introduction

Augmentative and Alternative Communication (AAC) is used by disabled people to help them to communicate in daily life and to be more independent in their interactions with others (Beukelman and Mirenda, 1998). As such, AAC technologies also improve the social inclusion of disabled people, including those with an Intellectual Disability (ID), which are the focus of this paper.

One of the characteristics of AAC is to represent the natural language in the form of pictures or pictographs, to support the communication of people with language impairment. Several sets of pictographs have been designed specifically for people with an ID to express their basic needs to their family, friends, teachers, or healthcare professionals. Pictures can be used by these persons for communication in various situations such as for social media (Vandeghinste et al., 2015), access to school (Vaschalde et al., 2018), or in the (pre-)hospital setting (Vaz, 2013; Eadie et al., 2013).

Recent research focuses on applications with pictographs for medical settings and people with or without disabilities such as foreigners and allophone patients. This is the case for the Smartwatch prototype of Wolk et al. (2017) and the BabelDr project (Bouillon et al., 2017; Norré et al., 2021a,b). Current systems are often limited and do not always use NLP techniques, especially the applications available for the general public. My Symptoms Translator (Alvarez, 2014; Alvarez and Fortier, 2014) and MediPicto AP-HP1 are examples of mobile app for medical communication with images between doctors and patients.

This article focuses on the Text-to-Picto system, which automatically translates text into pictographs for people with an ID (Sevens, 2018; Vandeghinste et al., 2015). The system was originally designed for Dutch, and later extended to English and Spanish (Sevens et al., 2015). In this work, we adapt it to French. In addition, we extend the system by linking it to a third pictograph set, namely Arasaac2. Until now, two pictograph sets had been used in Text-to-Picto: Sclera3 and Beta4, but adding Arasaac was relevant in view of its growing popularity and coverage (more than 15,000 coloured pictographs, which are specifically designed for people with an ID).

This paper first refers to some related work (Section 2), before introducing the methodology used to adapt Text-to-Picto to French (Section 3). Then, we automatically and manually evaluate the French translation system with the three pictograph sets, using three corpora corresponding to different use cases of AAC. Results are also compared to those of similar systems (Section 4). Finally, we discuss the different evaluations of the system (Section 5).

1https://www.aphp.fr/medipicto
2https://arasaac.org
3http://www.sclera.be
4http://www.betasymbols.com
2 Related Work

In this section, we present some work about text-to-pictograph translation systems integrating various NLP techniques for different languages.

In their translation system, Mihalcea and Leong (2008) used the WordNet resource (Miller, 1995), but without exploiting the relations between concepts. In addition, their system aimed to translate only the content words (nouns and verbs). The Glyph automatically translated patient instructions using NLP (e.g. preprocessing stage, including sentence splitter, word and synonym normalization, etc.), terminology or medication databases, but also computer graphics techniques (Zeng-Treitler et al., 2014; Bui et al., 2012). This application was not designed or tested with disabled people unlike the work of Sevens (2018) and Vandeghinste et al. (2015) on the Dutch Text-to-Picto system, later extended to English and Spanish in the framework of the Able to Include project. The Text-to-Picto system had a certain success: Kultsova et al. (2017) integrated it in an assistive mobile application for travel and communication in Russian, intended for people with an ID, whereas Nandy (2019) adapted it for Indian languages. Other systems were also developed recently such as AraTraductor, an application for Spanish using NLP to improve its pictograph translations (Bautista et al., 2017) or the system of Imam et al. (2019) for English, which uses WordNet and ImageNet (Deng et al., 2009).

For French, Vaschalde et al. (2018) implemented a speech-to-picto tool with an automatic speech recognition module. It includes a system to automatically translate text into Arasaac pictographs, the set of images for AAC used for this study. Based on the work of Vandeghinste et al. (2015), Vaschalde et al. (2018) also evaluated their prototype by testing word sense disambiguation and automatic simplification for the passive structures or the deletion of some grammatical words. All these techniques were already tested by Sevens (2018), but only for translation from Dutch. Sevens et al. (2017) developed a rule-based module for simplification of twelve syntactic phenomena (relative clause, non subject-verb-object order, etc.), including compression. For pictograph translation, Sevens et al. (2016) also used a Dutch word sense disambiguation tool. It is worth mentioning that for the French language, there are not many large resources (e.g. sense-annotated corpus) – compared for example to English – for these NLP tasks.

3 Methodology

In this section, we describe how we automatically linked the pictographs to lexical-semantic resources. The Princeton WordNet or PWN (Miller, 1995) is one of the largest lexical databases for English. It classifies verbs, nouns, adjectives and adverbs into sets of cognitive synonyms, called synsets, which are linked by semantic relations. Its latest versions are the PWN 3.0 and the PWN 3.1, whose synsets do not have the same numeric identifiers.

For French, we found the WOrdnet Libre du Francais or WOLF (Sagot and Fisler, 2008) and the WoNeF (Pradet et al., 2014), two automatic translations of PWN 3.0 that differ in the way they were built. In this work, we use the WOLF 1.0b4 (2014) and the three versions of WoNeF 0.1 (2012): coverage (c), fscore (f), and precision (p). The WOLF is considered as the standard French WordNet and is cited more often than WoNeF. Compared to the 117,659 synsets of PWN 3.0, it contains 56,475 synsets with at least one lemma translated into French (see Table 1). As regards the WoNeF, the high coverage version contains 109,447 pairs (literal, synset), the main WoNeF has a F-score of 70.9%, and the high precision version has a precision of 93.3%. In addition, as a result of optimizing the three metrics, the coverage version includes 55,697 synsets, the fscore version has 53,440 and the precision has only 15,482 (Pradet et al., 2014).

<table>
<thead>
<tr>
<th></th>
<th>WOLF 1.0b4</th>
<th>WoNeF 0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>c</td>
<td>f</td>
</tr>
<tr>
<td>N</td>
<td>42,427</td>
<td>37,685</td>
</tr>
<tr>
<td></td>
<td>41.66%</td>
<td>45.89%</td>
</tr>
<tr>
<td>V</td>
<td>5,870</td>
<td>5,772</td>
</tr>
<tr>
<td></td>
<td>42.63%</td>
<td>41.92%</td>
</tr>
<tr>
<td>ADJ</td>
<td>6,691</td>
<td>10,238</td>
</tr>
<tr>
<td></td>
<td>36.85%</td>
<td>56.38%</td>
</tr>
<tr>
<td>ADV</td>
<td>1,487</td>
<td>2,002</td>
</tr>
<tr>
<td></td>
<td>41.06%</td>
<td>55.28%</td>
</tr>
<tr>
<td>Total</td>
<td>56,475</td>
<td>53,440</td>
</tr>
</tbody>
</table>

Table 1: Number of non-empty synsets of French WordNets and percentage compared to PWN 3.0 per POS.
The Text-to-Picto system was designed to be language independent and easily extensible to other languages. For the Dutch version, Vandeghinste and Schuurman (2014) had manually linked 5,710 Sclera pictographs and 2,760 Beta pictographs to Cornetto WordNet before linking them automatically to PWN 3.0 for English and to MCR WordNet 3.0 for Spanish (Sevens et al., 2015).

To extend the Text-to-Picto system to French and to the Arasaac pictograph set, we could not get access to the links between the PWN 3.0 and 800 Arasaac pictographs of Schwab et al. (2020). We therefore used the Arasaac API to get JSON data, including the manual links to the PWN 3.1, different pictograph filenames (i.e. the lemmas), and their numeric identifiers (allowing to access the pictograph url). The pictograph filenames are available for at least 30 languages, including English, Spanish, Dutch, Russian, Arabic, etc. The other data of Arasaac API are the same for all languages.

Using OpenRefine (Verborgh and De Wilde, 2013), we cleaned the data: sort, deletion, preprocess duplicate filenames (renamed by adding numbers to distinguish the pictographs in the reference corpora), etc. We automatically linked the Arasaac pictographs associated with one or more synsets of PWN 3.1, WOLF and WoNeF through the PWN 3.0 identifiers and the Collaborative Interlingual Indexes (CILI) available on the GitHub repository of the Global WordNet Association. It would also be possible to have translations from the Open English WordNet (McCrae et al., 2020) into Arasaac pictographs. This WordNet and the PWN 3.1 have the same synset identifiers.

For example (see Figure 1), the Arasaac pictograph docteur or médecin (doctor) has the identifier 2467. It is associated with a PWN 3.1 synset and other information (e.g. one or several tags) that we separated in other tables for future work. By transferring the English synset automatically with CILI, we obtain the PWN 3.0 synset, POS, relation(s) and lemma(s) of French WordNets. In this case, {docteur/médecin/toubib} for WOLF and only {médecin} for WoNeF (coverage and fscore versions) in which the lemmas {docteur} and {toubib} are linked to another synset. Surprisingly, in the WoNeF (precision), there is no lemma {docteur} and {médecin} although they are frequent terms for doctor, but only the rare term {toubib}, linked to two other synsets.

![Figure 1: Mapping Arasaac pictographs with French WordNets.](image-url)

As a result of this process, our Text-to-Picto system is the first that uses synsets of the French WordNets. In contrast, Schwab et al. (2020) directly used the PWN 3.0 and automatically translated the text because they consider the original PWN as the most complete and reliable database. As mentioned before, the automatic translations of WOLF and WoNeF versions differ and are, indeed, less complete compared to PWN 3.0.

### 3.2 Description of the System

We describe the architecture of our system used to translate a textual input into a sequence of Sclera, Beta, or Arasaac pictographs (see Figure 2).

![Figure 2: Architecture of the Text-to-Picto system adapted from Vandeghinste et al. (2015).](image-url)

The source text first undergoes shallow linguistic analysis: on the one hand, sentence detection, tokenization, part-of-speech tagging and lemmatization,
tion are carried out by TreeTagger (Schmid, 1994); on the other hand, we added detection of Multi-Word Expressions (MWE), processing of specific French phenomena (e.g. elision, negation variants), and simple Named Entity Recognition (NER) based on rules and dictionaries. As in Vaschalde (2018), the named entities detected are substituted by generic placeholders such as character or city.

For example (see Figure 3), the sentence Max ira à Leuven l’été, au revoir (Max will go to Leuven the summer, goodbye) is translated by a sequence of seven pictographs perso, aller, à, ville, le, été, au revoir (character, go, to, city, the, summer, goodbye) after the shallow linguistic analysis. Without the rules of elision and MWE detection, the article le (the) would not be translated and the MWE au revoir (goodbye) would be incorrectly translated, i.e. by two pictographs: the preposition au (at) and the verb revoir or corriger (revise).

In the next step, two routes are possible depending on the word to translate: the semantic route and the direct route. In the semantic route, each word is looked up in the WordNet database. In case a word is not found, we leverage two WordNet relations – has_hyperonym and near_antonym – to get substitute translation. For example, there is no pictograph for saumon (salmon), the word is therefore translated by its hyperonym poisson (fish). The word infecter (infect) does not have a pictograph either and is translated by its antonym followed by the negative pictograph, i.e. désinfecter pas (desinfect no).

In their annotations for Sclera and Beta sets, Vandeghinste and Schuurman (2014) indicated whether the pictograph is complex or not, i.e. whether it represents several concepts (verb + noun, noun + noun, noun + adjective). For example, manger un sandwich (eat a sandwich) is translated by the single pictograph boterham-eten in Sclera. This filename is linked to the head synset {manger/alimenter/déjeuner} (eat/feed/lunch) and to the dependent synset {sandwich}. Information about pictographs corresponding to a MWE is missing for Arasaac. It is worth mentioning that inflection is taken into account for MWE annotated with two synsets, unlike the MWE detection used in the shallow linguistic analysis.

For the direct route, we build a dictionary for each of our three pictograph sets for the words not covered by WordNet, i.e. pronouns, prepositions, etc. In Sclera and Beta, the pictographs were linked to their Dutch filename by Vandeghinste et al. (2015). We have then manually translated these two Dutch dictionaries into French. For Arasaac, pictographs were manually linked to French lemmas through their identifiers. A part-of-speech tag were also used to distinguish certain homonyms, e.g. the negative adverb pas (not) and the noun pas (step). As a result, our dictionaries provide respectively 412, 298 and 420 direct links between pictographs and French tokens or lemmas.

To choose the optimal path while converting a sequence of lemmas to a sequence of pictographs, we use the search algorithm A* described in detail by Vandeghinste et al. (2015). It works with different parameters (i.e. penalties) related to WordNet relations, pictograph features and route preference.

### 3.3 Use Cases

We briefly describe our three corpora representing several use cases of AAC. They are used for the automated and manual evaluation of our system in which we use different metrics to compare the system’s output to a reference translation.
1. The **Email Corpus** (130 sentences), manually translated into Sclera and Beta pictographs by Sevens (2018). The emails, written by people with an ID, their teachers, or their parents, were extracted from the WAI-NOT Belgian website. We manually translated this Dutch corpus into French and Arasaac pictographs. We have slightly pre-edited the reference translations into Sclera and Beta to maintain French word order of our corpus. We did not reproduce the spelling mistakes of people with an ID because we do not evaluate the automated spelling correction.

2. The **Book Corpus** (254 sentences), consisting of six copyright-free children stories manually translated into Arasaac pictographs by Vaschalde (2018). We have slightly pre-edited this French corpus to make it compliant with our evaluation format: e.g., we did not translate plural words twice as is the case in Vaschalde (2018). For example, *les couvertures* (the covers) in its source corpus is manually translated by *le couverture couverture* (the cover cover) in its reference corpus. In our reference corpus, we replaced it by *les couverture* without repetition and keeping the plural for the article because we added the pictograph *les* in our dictionary. We also modified some filenames in their reference translations because we renamed the duplicates when we preprocessed the Arasaac data.

3. The **Medical Corpus** (260 sentences), is a subset from BabelDr, a medical translation system (Bouillon et al., 2017). These sentences are relatively simple compared to some variations offered by the system. There are mainly questions from doctors to patients, i.e. *pouvez-vous décrire la douleur ?* (can you describe the pain?) or *à combien était votre température la dernière fois que vous l’avez mesurée ?* (was what your temperature the last time you measured it?). There are also patient instructions such as *je vais m’occuper de vous aujourd’hui* (I will take care of you today). As for the Email Corpus, we manually translated the Medical Corpus into Arasaac pictographs. The Figure 4 shows an example of reference translation, the sequence of five filenames: *avoir, vous, des, carie, ?* (have, you, the, cavities, ?).

As regards manual translations, all the words in the source text are translated into pictograph filenames in our reference corpora. However, the process is not a literal translation. We have sometimes translated several words into a single pictograph: e.g. for MWE such as *bouteille de coca* (bottle of coca-cola) or *envoyer une lettre* (send a letter), which can be translated by the complex pictograph *Coca-Cola* or *envoyer*2 in Arasaac (see Figure 5). Some Arasaac pictographs can also have different filenames or meanings, e.g. the pictograph for *mal de tête* (headache) or *faire mal* (hurt).

![Figure 5: Example of complex pictographs for bouteille de coca (Sclera/Arasaac), envoyer une lettre (Sclera/Arasaac) and mal de tête or faire mal (Arasaac).](image)

4 Results

This section presents how we tuned the system (Section 4.1) and describes the results of the automated evaluation (Section 4.2), followed by the manual evaluation (Section 4.3).

4.1 Tuning the System

For tuning our system, we used 56 sentences sampled from the Email Corpus, that is our development set. We should also stressed that running several times the Text-to-Picto system on the same sentence may yield slightly different translations, even with the same parameters (cf. Section 5). Therefore, each BLEU score has been computed as the average over 10 runs (translations). We also report the standard deviation (SD) over the 10 runs.

We first experimented with the WordNets: the WOLF and the three versions of WoNeF – coverage (c), f-score (f), and precision (p). At this step, as we did not know the optimal parameters for Text-to-Picto yet, we used the best ones reported by Sevens (2018) for Sclera and Beta sets (for Arasaac, we
took Beta’s parameters). The standard BLEU score (Papineni et al., 2002), reported in Table 2, allowed us to choose WOLF as the best French WordNet, as it obtains the highest BLEU scores regardless of the pictograph set. Therefore, WOLF will be used for all our evaluations. These results can be explained because WOLF is often connected to more synsets than the WoNeF; therefore it is more likely that there is a link to the pictograph. For WOLF, the SD is always higher than for WoNeF (SD equals 0 for the high precision version, as its small size makes it less likely to refer to several lemmas).

<table>
<thead>
<tr>
<th>Sclera</th>
<th>Beta</th>
<th>Arasaac</th>
</tr>
</thead>
<tbody>
<tr>
<td>WOLF</td>
<td>25.9 (0.36)</td>
<td>25.8 (1.42)</td>
</tr>
<tr>
<td>WoNeF c</td>
<td>13.7 (0.10)</td>
<td>17.5 (0.26)</td>
</tr>
<tr>
<td>WoNeF f</td>
<td>13.7 (0.09)</td>
<td>17.6 (0.19)</td>
</tr>
<tr>
<td>WoNeF p</td>
<td>12.4 (0.00)</td>
<td>12.2 (0.00)</td>
</tr>
</tbody>
</table>

Table 2: Results of the French Text-to-Picto for WOLF and WoNeF by pictograph set with BLEU metric.

In the next step, we tuned the parameters (cf. Section 3.2) through an automated procedure, using a local hill climbing algorithm (Vandeghinste et al., 2015) with BLEU as the evaluation metric. For each pictograph set – Sclera (S), Beta (B) and Arasaac (A) –, we run five trials of 50 iterations with different random initialisation of the parameters and using a granularity of one, in order to cover different areas of the search space. Finally, we took the best scoring parameter values (see Table 3).

<table>
<thead>
<tr>
<th>WOLF relations</th>
<th>Min</th>
<th>Max</th>
<th>S</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threshold</td>
<td>5</td>
<td>20</td>
<td>11</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>Hyponym penalty</td>
<td>0</td>
<td>15</td>
<td>13</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>Antonym penalty</td>
<td>0</td>
<td>15</td>
<td>6</td>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pictograph features</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Wrong number</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>No number</td>
<td>0</td>
<td>10</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Route preference</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Out-Of-Vocabulary</td>
<td>0</td>
<td>10</td>
<td>5</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Direct route advantage</td>
<td>0</td>
<td>15</td>
<td>12</td>
<td>5</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 3: Results of the parameter tuning of French Text-to-Picto for Sclera, Beta and Arasaac pictograph sets.

4.2 Automated Evaluation

We automatically evaluated the performance of our French Text-to-Picto system on the Email Corpus, the Book Corpus and the Medical Corpus. Different experimental conditions were tested, progressively activating more features of the system: a) only with dictionary, b) with dictionary and synonyms of WOLF, c) with dictionary, synonyms and other relations of WOLF (i.e. hyperonyms, antonyms). In addition, we compared our results with those of the Dutch Text-to-Picto system (Sevens, 2018) and those of the French system of Vaschalde et al. (2018), when available. Such comparisons should be taken with caution, as the experiments are not strictly comparable.

For a better comparison with these studies, we reused the metrics of Sevens (2018) and Vandeghinste et al. (2015): the BLEU (Papineni et al., 2002),9 the Word Error Rate (WER) and the Position-independent word Error Rate (PER). BLEU is designed for machine translation, while WER and PER are used in speech recognition. All of these metrics compare the system output to one or more reference translations. In our case, we used sentences translated manually into pictograph filenames (cf. Section 3.3). As described in Section 4.1, each evaluation metric was estimated based on an average over 10 runs of the system. Standard deviations are reported in brackets in the tables.10

<table>
<thead>
<tr>
<th>Sclera</th>
<th>Beta</th>
<th>Arasaac</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLEU</td>
<td></td>
<td>WER</td>
</tr>
<tr>
<td>Dictionary</td>
<td>12.0 (0.0)</td>
<td>58.7 (0.0)</td>
</tr>
<tr>
<td>(Sevens, 2018)</td>
<td>14.1</td>
<td>71.9</td>
</tr>
<tr>
<td>+ Synonyms</td>
<td>17.8 (0.4)</td>
<td>56.2 (0.4)</td>
</tr>
<tr>
<td>(Sevens, 2018)</td>
<td>16.5</td>
<td>67.5</td>
</tr>
<tr>
<td>+ Relations</td>
<td>17.9 (0.4)</td>
<td>56.2 (0.3)</td>
</tr>
<tr>
<td>(Sevens, 2018)</td>
<td>16.1</td>
<td>68.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Beta</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary</td>
<td>10.9 (0.0)</td>
<td>63.0 (0.0)</td>
</tr>
<tr>
<td>(Sevens, 2018)</td>
<td>16.9</td>
<td>63.4</td>
</tr>
<tr>
<td>+ Synonyms</td>
<td>21.6 (1.4)</td>
<td>57.5 (1.1)</td>
</tr>
<tr>
<td>(Sevens, 2018)</td>
<td>23.0</td>
<td>52.4</td>
</tr>
<tr>
<td>+ Relations</td>
<td>22.4 (1.2)</td>
<td>57.9 (0.5)</td>
</tr>
<tr>
<td>(Sevens, 2018)</td>
<td>25.9</td>
<td>51.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arasaac</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLEU</td>
</tr>
<tr>
<td>Dictionary</td>
</tr>
<tr>
<td>(Sevens, 2018)</td>
</tr>
<tr>
<td>+ Synonyms</td>
</tr>
<tr>
<td>(Sevens, 2018)</td>
</tr>
<tr>
<td>+ Relations</td>
</tr>
<tr>
<td>(Sevens, 2018)</td>
</tr>
</tbody>
</table>

Table 4: Results of the French Text-to-Picto and Dutch Text-to-Picto on Email Corpus by pictograph set with BLEU, WER and PER metrics.

We compared our results with those of Sevens (2018) for Dutch on the same 84 sentences from the Email Corpus (see Table 4). As regards BLEU scores, we got very comparable results, especially

---

9The standard BLEU based on average of 1-grams, 2-grams, 3-grams, and 4-grams (mteval-v011b.pl on https://github.com/moses-smt/mosesdecoder).

10For the first condition (a), the SD is always 0 because, as without WordNet, there is no variation in the results.
for Sclera. For Beta, our scores are lower than those of Sevens (2018). Arasaac obtains the best BLEU scores, except for the first condition (a). This can be explained by the fact that dictionaries for Sclera and Beta are more suitable for this task. They were built from frequent untranslated words from the Email Corpus. For the conditions (b) and (c), differences does not seem meaningful.

For the Book Corpus, as proper names occur frequently, we added one condition: d) without the step of Named Entity Recognition (NER), based on the work of Vaschalde (2018). Results are reported at Table 5. Our BLEU scores for conditions (b) (28.0) and (c) (28.3) are in line with those of Vaschalde et al. (2018). Their translation system from French into Arasaac pictographs obtains a BLEU score of 25.45 when all the words are translated, without word sense disambiguation and without a specific treatment for plurals (26.65 with it). The rather decent score of the dictionary condition (a) on this corpus – in contrast with its performance on the two other use cases – can probably be explained by the effect of the NER module. Indeed, the test set includes 152 occurrences of proper names and we observe that, without NER module (d), scores drop a lot.

<table>
<thead>
<tr>
<th>Condition</th>
<th>BLEU</th>
<th>WER</th>
<th>PER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary</td>
<td>18.0 (0.0)</td>
<td>49.8 (0.0)</td>
<td>48.5 (0.0)</td>
</tr>
<tr>
<td>+ Synonyms</td>
<td>28.0 (0.4)</td>
<td>58.0 (0.9)</td>
<td>49.8 (0.8)</td>
</tr>
<tr>
<td>+ Relations</td>
<td><strong>28.3</strong> (0.6)</td>
<td>57.7 (0.7)</td>
<td>49.3 (0.6)</td>
</tr>
<tr>
<td>- NER</td>
<td>17.4 (0.7)</td>
<td>68.1 (1.1)</td>
<td>57.1 (0.8)</td>
</tr>
</tbody>
</table>

Table 5: Results of the French Text-to-Picto on Book Corpus for Arasaac pictograph set with BLEU, WER and PER metrics.

<table>
<thead>
<tr>
<th>System</th>
<th>BLEU</th>
<th>WER</th>
<th>PER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sclera (Sevens, 2018)</td>
<td>Email – Dutch</td>
<td>89.24%</td>
<td>86.23%</td>
</tr>
<tr>
<td></td>
<td>Email – English</td>
<td>93.30%</td>
<td>73.04%</td>
</tr>
<tr>
<td></td>
<td>Email – Spanish</td>
<td>93.31%</td>
<td>83.14%</td>
</tr>
<tr>
<td>Beta (Sevens, 2018)</td>
<td>Email – Dutch</td>
<td>85.91%</td>
<td>89.45%</td>
</tr>
<tr>
<td></td>
<td>Email – English</td>
<td>82.56%</td>
<td>86.14%</td>
</tr>
<tr>
<td></td>
<td>Email – Spanish</td>
<td>94.64%</td>
<td>86.83%</td>
</tr>
<tr>
<td>Arasaac</td>
<td>Medical – French</td>
<td>83.70%</td>
<td>90.14%</td>
</tr>
</tbody>
</table>

Table 7: Results of the French Text-to-Picto on Medical Corpus and Dutch/English/Spanish Text-to-Picto on Email Corpus by pictograph set with Precision, Recall and F-score metrics.

By comparison, Sevens (2018) obtained on its Email Corpus a F-score between 81-85% for English-to-Sclera/Beta system and 87-91% for Dutch/Spanish-to-Sclera/Beta (without counting proper names). Our French system obtains a higher recall than precision and recall of other linguistic versions. In our manual evaluation, we assume that all words must be translated.

5 Discussion

As regards the efficiency of the different pictograph sets, tested on the Email Corpus, we see that our results are better for the Beta set than for the Sclera set. Sevens (2018) explained that Beta contains less pictographs than Sclera. As a result, more paraphrasing translations are possible in Sclera, resulting in a less accurate measurement of translation quality by BLEU. However, our scores for Arasaac, the largest set of pictograph, are the best. Compared to others, this set includes more function words (articles, prepositions, etc.) that have been encoded in our dictionary. Therefore, they will always be well translated, which improves the results. It should also be mentioned that, despite
relatively good results, some pictographs generated by our system are not also easily comprehensible, depending on the context of the sentence. This is especially the case for function words and pain description on a specific body part. In addition, medical words are not always translated because there is no corresponding pictographs, e.g., symptôme (symptom), cancer (tumor), etc.

Our experiments also aimed at assessing the contribution of different components to our Text-to-Picto system for French. Using only the dictionary clearly yields unsatisfactory results. The system improves when we add the synonym or the relation component, regardless of the pictograph sets and corpora. However, the difference between both components appears marginal. In our reference corpora, a manual inspection of the data reveals that the relation of synonymy is much more frequent than those of hyperonymy and antonymy – which is very rare –, especially for the largest Arasaac pictograph set. Translating text into pictographs is a meticulous and time-intensive process (Sevens et al., 2016). This explains why the corpora are small. It is worth mentioning that the BLEU score is very dependent on the reference translation, which may be partially subjective.

Finally, as explained above, every time we run the French system with the same parameters, we get slightly different translations. This happens when the optimal path calculation step has to choose randomly between several pictographs that have an equal weight, using WordNet. For future work, it would be possible to associate the pictographs with frequency information to regulate this issue. Some studies (Imam et al., 2019; Sevens, 2018; Sevens et al., 2016) also showed that word sense disambiguation improves the results of text-to-picto systems. This would avoid translation errors related to homonyms identified in our manual evaluation, e.g., enceinte (to be pregnant or a speaker) and bleu (the colour or a bruise).

6 Conclusion

We presented the French version of the Text-to-Picto system, which automatically translates a textual input into pictographs for people with an ID. Our experiments show that this system is easily extensible to other natural or pictograph languages.\textsuperscript{11}

Compared to the Dutch version, we adapted the shallow linguistic analysis by adding new steps (detection of MWE, preprocessing of specific phenomena, and simple NER). Data cleaning was performed to link the Arasaac pictographs to French semantic resources. The evaluations on the Email and the Book Corpus with WOLF show that our results are indeed in line with those of previous studies. However, there is room for further improvement, for instance adding a word sense disambiguation step to select the right pictograph for a given meaning. We also carry out automated and manual evaluations on a new use case: medical data, which raised new challenges related to the translation of technical terms. We have seen above that our system currently tends to poorly handle technical terms, often missing from WOLF. We plan to investigate solutions to this limitation, for example by applying automatic text simplification for the medical domain (Cardon and Grabar, 2020) on the original sentences.

We also plan to run tests with target users to tune this Text-to-Picto system for medical communication between doctors and patients in the hospital setting. The Dutch version of the presented system has already been tested in real situations with a focus group of five adults with an ID and two coaches in a day centre in Belgium (Sevens, 2018).

Acknowledgments

This research was funded by the UCL-FSR mandate N°13936.2020. This work is also part of the PROPICTO project, funded by the Fonds National Suisse (N°197864) and the Agence Nationale de la Recherche (ANR-20-CE93-0005). The pictographs used are property of the Aragon Government and have been created by Sergio Palao to Arasaac (http://arasaac.org). Aragon Government distributes them under Creative Commons License (BY-NC-SA). The other pictographs used are property of Sclera vzw (https://www.sclera.be/) which distributes them under Creative Commons License.

References


\textsuperscript{11}The source code of the system and the French corpora used for evaluation will be made available for the research community at the following address: https://github.com/VincentCCL/Picto.


Céline Vaschalde, Pauline Trial, Emmanuelle Esperança-Rodier, Didier Schwab, and Benjamin Lecouteux. 2018. Automatic pictogram generation from speech to help the implementation of a mediated communication. In *Conference on Barrier-free Communication*.


Transfer-based Enrichment of a Hungarian Named Entity Dataset

Attila Novák and Borbála Novák
MTA-PPKE Hungarian Language Technology Research Group,
Pázmány Péter Catholic University
Faculty of Information Technology and Bionics
Práter u. 50/a, 1083 Budapest, Hungary
\{surname.firstname\}@itk.ppke.hu

Abstract

In this paper, we present a major update to the first Hungarian named entity dataset, the Szeged NER corpus. We used zero-shot cross-lingual transfer to initialize the enrichment of entity types annotated in the corpus using three neural NER models: two of them based on the English OntoNotes corpus and one based on the Czech Named Entity Corpus fine-tuned from multilingual neural language models. The output of the models was automatically merged with the original NER annotation, and automatically and manually corrected and further enriched with additional annotation, like qualifiers for various entity types. We present the evaluation of the zero-shot performance of the two OntoNotes-based models and a transformer-based new NER model trained on the training part of the final corpus. We release the corpus and the trained model.

1 Introduction

1.1 Resources

Named entity recognition is a fundamental NLP task that has played an important role in tasks like information extraction, document deidentification, conversational models, etc. Following the annotation scheme used in the CoNLL 2002/2003 NER annotation tasks, legacy named entity corpora usually contain annotation of four entity types: organizations (ORG), persons (PER), locations (LOC) and general entity category covering all the rest (MISC). This is the case for all named entity corpora available for Hungarian, the Szeged NER corpus (Szarvas et al., 2006), the silver-standard Hungarian hunNERwiki corpus (Simon and Nemeskey, 2012) automatically derived from Wikipedia, and the recently published NerKor corpus.\footnote{https://github.com/nytud/NYTK-NerKor} The English OntoNotes 5 corpus (Weischedel et al., 2013), on the other hand, contains a richer set of entities. Geopolitical entities (GPE: countries, settlements, etc.) and facilities (FAC: buildings, roads, airports etc.) are differentiated from geographical locations like continents or bodies of waters. Within the MISC category, products (PROD), laws and other norms (LAW), events (EVENT) and titles of works of art (WORK_OF_ART) are differentiated. In addition, the OntoNotes NER tagset also encompasses time and numerical expressions distinguishing dates and times, cardinal and ordinal numbers, quantities, percentages and amounts of money. In addition, other categories covering non-entities like languages (LANGUAGE) and nationalities, religions and political affiliations (NORP ‘nationality/other/religion/political’) are covered, presumably just because English orthography happens to prescribe capitalization for words (adjectives in the case of NORP) belonging to this category.

Some resources in languages other than English also use NER tagsets richer than the basic four-class tagset. Although the NoSta-D resource used in the GermEval2014 shared task targeting German NER (Benikova et al., 2014) maintains a four-class distinction, words (especially adjectives) derived from names as well as compounds containing them are marked as such. This corpus, similarly to other resources like the GENIA corpus (Kim et al., 2003) containing biomedical entities and the Spanish and Catalan newspaper text corpus AnCora (Taulé et al., 2008), also features nested named entities. One of the most richly annotated NER corpora is the Czech Named Entity Corpus (Ševčíková et al., 2007). It contains both a rather rich set of entity types and nested entities.

1.2 Architectures for Sequence Tagging and Cross-lingual Transfer

Legacy data-driven statistical machine learning algorithms based on Hidden Markov Models (Baum
and Petrie, 1966), Maximum Entropy models (Ratnaparkhi, 1996) and CRF (Lafferty et al., 2001) provided then state-of-the-art performance for sequence tagging, however they relied on data and features pertaining strictly to the target language. This meant that a significant amount of annotated training data in the target language was required to attain acceptable performance using these models.

The paradigm shift to neural models offered the possibility of changing this situation. Already the simplest non-contextual distributional word embedding models like word2vec (Mikolov et al., 2013a,c) were discovered to have some kind of inherent language-independent property. It was found that models trained on different languages independently can be mapped to each other with high accuracy using a rather limited bilingual vocabulary (Luong et al., 2015) or even in an unsupervised manner (Mikolov et al., 2013b).

It was also discovered that, with neural machine translation models, it is possible to improve performance in specific lower-resource languages simply by training the encoder and the decoder of the model in a shared manner on multiple languages. This resource-sharing also made direct translation between all of the represented languages possible, and resulted in savings in resources concerning both training, storage and inference, i.e. using the model in production. The models offering state-of-the-art performance in machine translation performed similarly well in other NLP tasks. Pre-training the encoder of models used in NMT (especially the now-ubiquitous transformer architecture (Vaswani et al., 2017)) for simple mask-filling tasks on high amounts of (monolingual) plain text resulted in contextual language models that could be fine-tuned for specific tasks in a much more efficient manner than training similar models from scratch (Devlin et al., 2019). These models significantly improved the state-of-the-art for nearly all NLP-related tasks even for high-resource languages like English. The improvement is even more significant in the case of lower-resource languages.

Multilingual training turned out to be fruitful not only in the domain of machine translation. The publication of multilingual contextual language models like multilingual BERT (Devlin et al., 2019) and XLM-RoBERTa (Conneau et al., 2019) made cross-lingual knowledge transfer efficient for other NLP tasks as well. It is possible to fine-tune the language model for e.g a token classification task, like named entity recognition in one language and apply it to another language. Even in a zero-shot scenario, where the token classification model has not seen any training data in the target language, it can provide a reasonable performance, especially if the target language is included among the languages covered by the underlying language model.

On the other hand, models trained on multiple languages were found not to provide state-of-the-art performance if a significant amount of training data is available for the given task in the target language. Fine-tuning a monolingual language model for a specific task usually results in better performance than using a heavily multilingual model like multilingual BERT, because the target language is usually relatively underrepresented in the underlying multilingual model (Martin et al., 2020). In this paper, we present a data annotation scenario in which we used zero-shot transfer to preannotate a language resource, which was then manually corrected and enriched to create a resource that can then be used to train a monolingual model to optimize performance.

2 Method

In the project presented in this paper, we significantly enriched the annotation in the first Hungarian named entity dataset, the Szeged NER corpus (Szarvas et al., 2006).

2.1 Zero-shot Preannotation

When preannotating the corpus, we fed the tokens to two models trained on the English OntoNotes 5 NER corpus. The first model was created by the DeepPavlov team fine-tuning multilingual BERT (Burtsev et al., 2018). The other model is based on XLM-RoBERTa, a multilingual contextual language model trained on a significantly bigger multilingual corpus than multi-BERT. The latter model is part of the FLAIR tool set (Akbik et al., 2019).

The two models use different tokenization following the tokenization scheme of the underlying contextual language model. The token sequence in the output of the models was thus different from the original input token sequence. This had to be taken into account when merging the annotation from the models with the original annotation. The merging procedure was automatic. While merging the annotations, in the case of overlapping entity spans, we considered the spans in the input anno-
tations gold standard, and if the zero shot model suggested a compatible entity subtype, we updated the entity type. E.g. an entity of type location (LOC) in the original annotation is compatible with any of geographical location (LOC), facility (FAC) and geopolitical entity (GPE). Annotation of non-entities, like dates, quantities and nationalities not present in the original annotation were introduced based on the output of the models.

2.2 Error Analysis and Automatic Error Correction

We identified typical errors of the zero-shot models that could be corrected automatically using regular-expression-based patterns. We have found that, in the case of transfer from English to Hungarian, a typical problem is that for many named entity types like names of organizations, bodies of waters, titles of works of art etc., a definite article is present in most but not all cases in Hungarian, while there is no article in English. This resulted in the model including definite articles for these types of entities in the annotation, an error that could be easily eliminated from the output.

While cross-lingual mapping resulted in some anomalies like inclusion of definite articles, it had other side-effects that we found to be useful. The output of the models also included annotation for adjectives derived from named entities like londoni ‘of London’. In contrast to the German NoSta-D corpus, words like this remained unannotated in all legacy Hungarian named entity corpora in spite of the fact that the identification of these words as references to named entities would be desirable in practical applications like information retrieval. We thus decided to keep this kind of annotation as part of our annotation enrichment effort.

After automatic correction of entity spans and types, we manually merged the outputs of the two models by checking the differences of the two annotations.

2.3 Considering a Third Model

We also applied a third model to the corpus. We used the Czech model of the NameTag 2 neural named entity tagger (Straková et al., 2019) trained on the Czech Named Entity Corpus CNEC 2 (Ševčíková et al., 2007). The underlying corpus and thus also the model contains a very fine-grained set of entity classes offering many subclasses within the broader categories like a distinction of companies vs. governmental/political institutions vs. academic/educational/cultural institutions and conferences/contests (the latter are also considered a subclass of organizations). NameTag 2 is capable of returning nested annotations (with a maximal depth of two overlapping entities). The model can be accessed via a web service. However, at least in the zero-shot cross-lingual setting, the annotation generated by this model seemed to be less accurate than those generated by OntoNotes-based models. Since there are no definite articles in Czech, this model had a similar problem including definite articles for the types of entities (e.g. organizations) that often appear with a definite article in Hungarian. It often generated two overlapping annotations for these types of entities differing only in whether the article is included. More importantly, the different occurrences of the same entity were often assigned different classes (usually this was an error rather than real ambiguity due to metonymic use). Also the extent of the span of the entities was less accurate than in the annotation generated by the English-based models. The subclassification itself also introduces problems of its own. It is not clear where sports clubs or central banks like the Bank of England should belong in this taxonomy.

2.4 Introduction of New Entity Types

Nevertheless, we found good use of the annotation generated by NameTag 2. As Hungarian is an agglutinating language and thus words appear in many different suffixed forms in the corpus, we applied lemmatization to the entity annotations generated by all models and aggregated the results listing the frequency of alternative annotations for the same entity. Tags generated by the FLAIR OntoNotes model and NameTag2 for the most frequent organizations in the corpus are shown in Table 1. Tags containing a hyphen in columns 3 to 8 were assigned by NameTag2, the rest by the FLAIR OntoNotes model. The list features the Budapest Stock Exchange (Budapesti Értéktőzsde = BÉT), the Budapest Commodity Exchange (Budapesti Árutőzsde), Nasdaq, Wall Street, the Hungarian Central Bank (Magyar Nemzeti Bank) and news agencies (MTI, MTI-ECO, Reuters). It is obvious that NameTag 2 struggles trying to assign them the right category. We thus refrained from adopting the taxonomy in CNEC 2.

On the other hand, the automatically generated gazetteer helped us identify entities really belonging to certain well-distinguishable entity classes
Table 1: Most frequent organizations in the corpus with several different annotations generated by the NameTag 2 tagger (labels containing a hyphen).

We generated regular-expression-based automatic correction patterns using manually marked entries from this automatically generated gazetteer (covering also suffixed forms), and mass-corrected annotations using these patterns.

We also discovered that certain types of expressions are mistagged by the zero-shot models due to lack of distinction in the original underlying OntoNotes annotation. One such example was expressions referring to time durations like ‘for five minutes’ or ‘six-day-long’. While other types of quantities are annotated in the OntoNotes NER resource as quantities resulting in sensible annotation also for the Hungarian input, the model mistagged duration expressions as time or date, which should only refer to expressions anchored to the timeline. We thus introduced a new entity type DUR to annotate unanchored duration expressions, and the annotation of many occurrences of this type of expressions could also be automatically introduced/corrected. We also annotated relative date expressions like days of week.

The Szeged NER corpus consists of business news, and due to its genre, it contains many occurrences of certain entity types not covered by the OntoNotes NER tagset: e.g. names of securities and stock exchange indexes. We introduced new tags for these entity types. They were also easy to identify in the generated gazetteer. We also introduced a tag for media like newspapers, broadcasting services and online news portals but refrained from distinguishing subtypes. This type of entities are somewhat similar to but can easily be distinguished from books (covered by the work of art category). On the other hand, they also involve an entity of the organization type (the publisher/redaction).

2.4.1 Metonymic Use of Names

Metonymic use of entities like referring to countries or other geopolitical entities as actors is usually annotated according to the actual metonymic sense. In the recently published NerKor corpus annotated with the coarse-grained ORG-PER-LOC-MISC tagset, references to countries as actors like Germany invaded France are annotated as ORG rather than LOC. This kind of metonymy is completely productive for all types of geopolitical entities, and annotating them as such solves the problem in a more elegant way than what the coarse-grained tagset makes possible. Incidentally, this
specific sort of metonymy is less prevalent in the business news genre than in certain other genres. On the other hand, Wall Street, one of the top entries on the organizations list in Table 1, is an example of an expression typically used in a metonymic sense referring to the New York Stock Exchange (and related financial institutions).

2.4.2 Qualifiers and Relations

In addition to the annotation mentioned above, we introduced two more tags that could be used to annotate qualifiers of named entities: QUAL and REL. These were used in situations where the named entity had a nominal modifier like in Ante Vulin\textsubscript{PER} építész\textsubscript{QUAL} ‘architect Ante Vulin’ or it was part of an appositive structure like in Jurij Lvov\textsubscript{PER} pénzügyminiszter-helyettes\textsubscript{QUAL} Világimír Putyn\textsubscript{PER} elnök\textsubscript{QUAL} bizalmasa\textsubscript{REL} ‘Yuri Lvov\textsubscript{PER}, Deputy Minister of Finance\textsubscript{QUAL}, confidant\textsubscript{REL} of President\textsubscript{QUAL} Vladimir Putin\textsubscript{PER}’. REL was used in situations where the phrase expresses a relation between named entities (if both are part of the same noun phrase), QUAL for other modifiers. We used this type of annotation to mark nominal phrases that are not named entities but as qualifiers of named entities have the same type of reference as the related named entity. We pregenerated and later refined this annotation using syntactic dependency parses of the sentences. The dependency structure could be used to identify the modified entities and thus their type, e.g. that elnök ‘president’ is a qualifier of persons and being a confidant is a relation between persons.

2.5 Filtering and Manual Annotation

We also filtered the corpus for repetitive boilerplate-like content: we removed identical sentences and ones differing only in numerical/date expressions. After creating the preannotation using the zero-shot models, automatically merging them with the original annotation and applying pattern-based corrections, we used the INCEpTION annotation framework (Klie et al., 2018) to correct and augment the annotations. Two researchers and five MA students of theoretical linguistics participated in the manual annotation process. Each document was revised by at least two annotators. Curation and final processing of the results was performed by a single researcher. In the current version, we refrained from generating nested annotation although currently the development of nested entity classifiers gained momentum, and some open-source neural nested entity taggers are available e.g. (Shibuya and Hovy, 2020) or (Wang et al., 2020). Although these models have sub-SOTA performance on flat NER datasets (we also found Nametag 2 to generate much less accurate annotation than the OntoNotes 5-based models), we will consider updating the dataset to have nested entities in a possible future release of the corpus. On the other hand, this resulted in ambiguities concerning the extent of entity spans and types, especially with the introduction of tags that mark non-names like qualifiers, nationalities etc.

![Table 2: The distribution of entity types in the original corpus](image)

<table>
<thead>
<tr>
<th>Entity Type</th>
<th>Original Corpus</th>
<th>Final Corpus</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>1294</td>
<td>225972</td>
</tr>
<tr>
<td>MISC</td>
<td>1662</td>
<td>14467</td>
</tr>
<tr>
<td>ORG</td>
<td>10529</td>
<td></td>
</tr>
<tr>
<td>PER</td>
<td>982</td>
<td></td>
</tr>
</tbody>
</table>

2.6 Properties of the Corpus

The size of the corpus is 206722 tokens, smaller than the original corpus due to the removal of repeated boilerplate content. On the other hand, it contains 40158 annotated spans, almost 2.8 times as much as in the original version. The distribution of entity types in the original Szeged NER corpus and the final version is shown in Tables 2 and 3. We conflated relational qualifiers with non-relational qualifiers of the same entity type. The MISC category in the final corpus covers only abbreviations annotated in the original corpus as MISC not referring to named entities. ORG-inf denotes entity mentions that refer to unique entities like names but use some informal reference instead of a name. This includes e.g. references to US departments, the FED and other governmental organizations which are referred to in Hungarian as ministries, offices etc., to central banks, stock exchanges etc. Note that there are more GPE entities than there were LOC entities in the original version due to adjectival forms also annotated and EU annotated as GPE rather than ORG. Many MEDIA entities were also ORG in the original corpus.

Entity types have an obviously skewed distribution with organizations, dates, money, nationalities and percentages (including ratios) dominating due to the genre of the corpus, while some tags are rather underrepresented. We plan to address this issue by adding text in other genres to balance the
3 Models and Performance

We tested the zero-shot performance of the original OntoNotes-based models on the final corpus disregarding entity types not covered by the OntoNotes annotation. The FLAIR model achieves $F_1 = 75.20$ with a great proportion of the errors coming from erroneously included definite articles. Considering all tag types, the performance is $F_1 = 67.46$. A simple fix of the definite article problem boosts performance on common tags to $F_1 = 87.91$, and $F_1 = 80.63$ on the full tagset.

The DeepPavlov model trained on the same dataset fared much worse achieving only $F_1 = 58.26$ on common tags and $F_1 = 53$ on the full tagset.

We trained a vanilla neural sequence tagger using the HuggingFace Transformers library (Wolf et al., 2020) fine-tuning the monolingual Hungarian huBERT language model (Nemeskey, 2021) using a 9:1 train:test split of the corpus. It achieved $F_1 = 92.69$, performing significantly better than the zero-shot models.

4 Conclusion

In this paper, we presented the procedure we followed to enrich the annotation in a legacy Hungarian NER resource by applying NER models based on multilingual language models and fine-tuned on NER corpora in other languages. We then made a significant effort identifying errors and correcting the annotation using automatic and semi-automatic methods, providing a solid base for the final manual annotation correction. We trained a neural sequence tagger on the final corpus achieving a solid $F_1 = 92.69$ performance.

Acknowledgments

This research was implemented with support provided by grants FK 125217 and PD 125216 of the National Research, Development and Innovation Office of Hungary financed under the FK 17 and PD 17 funding schemes as well as through the Artificial Intelligence National Excellence Program (grant no.: 2018-1.2.1-NKP-2018-00008).

References


One Size Does Not Fit All: Finding the Optimal Subword Sizes for FastText Models across Languages*

Vít Novotný
Faculty of Informatics
Masaryk University
witiko@mail.muni.cz

Eniafe Festus Ayetiran
Faculty of Informatics
Masaryk University
ayetiran@mail.muni.cz

Dalibor Bačovský
Faculty of Informatics
Masaryk University
456662@mail.muni.cz

Dávid Lupták
Faculty of Informatics
Masaryk University
dluptak@mail.muni.cz

Michal Štefánik
Faculty of Informatics
Masaryk University
stefanik.m@mail.muni.cz

Petr Sojka
Faculty of Informatics
Masaryk University
sojka@fi.muni.cz

Abstract
Unsupervised representation learning of words from large multilingual corpora is useful for downstream tasks such as word sense disambiguation, semantic text similarity, and information retrieval. The representation precision of log-bilinear fastText models is mostly due to their use of subword information.

In previous work, the optimization of fastText’s subword sizes has not been fully explored, and non-English fastText models were trained using subword sizes optimized for English and German word analogy tasks.

In our work, we find the optimal subword sizes on the English, German, Czech, Italian, Spanish, French, Hindi, Turkish, and Russian word analogy tasks. We then propose a simple n-gram coverage model and show that it predicts better-than-default subword sizes on the Spanish, French, Hindi, Turkish, and Russian word analogy tasks.

We show that the optimization of fastText’s subword sizes matters and results in a 14% improvement on the Czech word analogy task. We also show that expensive parameter optimization can be replaced by a simple n-gram coverage model that consistently improves the accuracy of fastText models on the word analogy tasks by up to 3% compared to the default subword sizes, and that it is within 1% accuracy of the optimal subword sizes.

1 Introduction

Bojanowski et al. (2017) have shown that taking word morphology into account is important for accurate continuous representations of words. However, they only show the optimal $n$-gram sizes on the German and English word analogy tasks (Bojanowski et al., 2017, Section 5.5). We continue their experiment by finding the optimal parameters on the Czech, Italian, Spanish, French, Hindi, Turkish, and Russian word analogy tasks and we show an up to 14% improvement in accuracy compared to the default subword sizes.

Furthermore, we propose a cheap and simple $n$-gram coverage model that can suggest near-optimal subword sizes for under-resourced languages, where the optimal subword sizes are unknown. We train our $n$-gram coverage model on the English, German, Czech, and Italian word analogy tasks, and we show that it suggests subword sizes that improve the accuracy by up to 3% on the Spanish, French, Hindi, Turkish, and Russian word analogy tasks and are within 1% accuracy of the optimal subword sizes on average. To make it easy for others to reproduce and build upon our work, we have publicly released a reference implementation of our $n$-gram coverage model.1

The rest of the paper is structured as follows: In Section 2, we discuss the related work. In Section 3, we discuss our methods and we propose our $n$-gram coverage model. In Section 4, we show and discuss our results. We conclude in Section 5 by summarizing our contribution. We outline the future work in Section 6.

2 Related work

Mikolov et al. (2013) described the Word2vec language model, which uses a shallow neural network to learn continuous representations of words: word embeddings. They also produced the English word analogy task, which tests how well word embeddings represent language regularities such as analogical relations (man is to woman what a king is to a queen), and evaluated Word2vec on their task.

1See https://github.com/MIR-MU/fasttext-optimizer.

Proceedings of Recent Advances in Natural Language Processing, pages 1072–1078
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_121
Berardi et al. (2015); Köper et al. (2015); Sloboda and Brychcín (2016); Cardellino (2019); Güngör and Yıldız (2017); Korogodina et al. (2020) produced the Italian, German, Czech, Spanish, Turkish, and Russian word analogy tasks for evaluating the performance of non-English word embeddings. Their findings revealed that, despite the morphological complexity of the languages, Word2vec language models can generate semantically and syntactically meaningful word embeddings.

In order to take word morphology into account, Bojanowski et al. (2017) developed the fastText language model based on Word2vec. Their improvements consisted of representing each word as a sequence of subwords with their own embeddings. They evaluated their models on the English, German, Czech, and Italian word analogy tasks. They also showed the optimal subword sizes of fastText on the English and German word analogy tasks. However, they did not optimize the subword sizes of fastText on the Czech and Italian word analogy tasks and used subwords of size 3–6\(^2\), which they described as “an arbitrary choice” (Bojanowski et al., 2017, Section 5.5).

Grave et al. (2018) produced the French and Hindi word analogy tasks. Furthermore, they also trained and publicly released fastText language models for 157 languages. Like Bojanowski et al., they also neglected to optimize the subword sizes. Unlike Bojanowski et al., they used subwords of size 5–5 for all languages, noting that “using character n-grams of size 5, instead of using the default range of 3–6, does not significantly decrease the accuracy (except for Czech).” (Grave et al., 2018, Section 4.3)

\[^2\]For subword sizes, we adopt the notation of Bojanowski et al. (2017) and Grave et al., 2018. For example, subwords of size 3–6 are all subwords whose size is 3, 4, 5, or 6.

### 3 Methods

In this section, we describe our methods and propose our n-gram coverage model, which can be used to suggest subword sizes for a fastText model without expensive parameter optimization.

#### 3.1 Optimal subword sizes

In the first part of our experiment, we train fastText language models on the English (22 GiB), German (8.3 GiB), Czech (1.2 GiB), Italian (4.2 GiB), Spanish (5.2 GiB), French (7.4 GiB), Hindi (0.57 GiB), Turkish (0.72 GiB), and Russian (9.9 GiB) Wikipedia corpora. We use subword sizes \(i–j\) for all \(i, j\), where \(1 \leq i \leq j \leq 10\), and we report the accuracies and the optimal subword sizes \(i–j\) on the English (Mikolov et al., 2013), German (Köper et al., 2015), Czech (Sloboda and Brychcín, 2016), Italian (Berardi et al., 2015), Spanish (Cardellino, 2019), French and Hindi (Grave et al., 2018), Turkish (Güngör and Yıldız, 2017), and Russian\(^3\) word analogy tasks.

#### 3.2 N-gram coverage

In the second part of our experiment, we compute and report the ratio between the frequencies of unique subwords of size \(i–j\) and the frequencies of all unique subwords of size less than 10 on the English, German, Czech, Italian, Spanish, French, Hindi, Turkish, and Russian Wikipedia corpora. In the following text, we call this ratio the n-gram coverage. Table 1 shows how the n-gram coverage is computed by example.

#### 3.3 Suggested subword sizes

In the third part of our experiment, we show that the n-gram coverage can be used to suggest subword sizes that are close to the optimal subword sizes on word analogy tasks.

\[^3\]See https://rusvectores.org/static/testsets/ru_analogy.txt.
For training, we compute the mean $n$-gram coverage for the optimal subword sizes on the English, German, Czech, and Italian word analogy tasks. For testing, we suggest subword sizes for the Spanish, French, Hindi, Turkish, and Russian word analogy tasks, so that the $n$-gram coverages for the suggested subword sizes on the testing word analogy tasks are the closest to the mean $n$-gram coverage for the optimal subword sizes on the training word analogy tasks. Notice that the suggested subword sizes are not based on the optimal subword sizes for the testing word analogy tasks.

After the performance estimation, we fold the training and testing word analogy tasks and we compute the mean $n$-gram coverage for the optimal subword sizes on all word analogy tasks (English, German, Czech, Italian, Spanish, French, Hindi, Turkish, and Russian). This means $n$-gram coverage can be used in applications of fastText to suggest the optimal subword sizes without expensive parameter optimization.

### 3.4 Language distances

In the final part of our experiment, we interpret suggested subword sizes as two-dimensional vectors and use the Euclidean distance to measure distances between languages. To see if our language distance measure represents interpretable linguistic phenomena, we compare it to the typological, geographical, and phylogenetic language distance measures of Littell et al. (2017):

#### Typological

Littell et al. define three typological language distance measures: syntactic, phonological, and inventory. Each distance measure is defined as the cosine distances between different feature vectors:

- The **syntactic** features describe the sentence structure of a language and have been adapted from the World Atlas of Language Structures (WALS), Syntactic Structures of World Languages, and Ethnologue databases.

- The **phonological** features describe the structure of the sound and sign systems of a language and have been adapted from the WALS and Ethnologue databases.

- The **inventory** features describe the presence or absence of distinctive speech sounds in the sound system of a language and have been adapted from the PHOIBLE database.

#### Geographical

The **geographical** language distance measure is defined as the cosine distance between feature vectors, where the features have been adapted from declarations of language location in the Glottolog, WALS, and SSWL databases.

#### Phylogenetic

The **phylogenetic** language distance measure is defined as the cosine distance between feature vectors, where the features correspond to the shared membership in language families, according to the world language family tree in the Glottolog database.

To compare our language distance measure with the language distance measures of Littell et al., we compute and report the Pearson’s correlation coefficient ($r$) between the distance measures.

### 3.5 Implementation details

We reproduce the experimental setup of Bojanowski et al. (2017, Section 4): skip-gram architecture, hash table bucket size $2 \cdot 10^6$, 300 vector dimensions, negative sampling loss with 5 negative samples, initial learning rate 0.05 with a linear decay to zero, sampling threshold $10^{-4}$, window size 5, and 5 epochs.

Like Bojanowski et al. (2017), we use a reduced vocabulary of the $2 \cdot 10^5$ most frequent words to solve word analogies. We use the implementation of word analogies in Gensim (Rehurek and Sojka, 2010), which uses Unicode upper-casing in the en_US.UTF-8 locale for caseless matching.

To compute Pearson’s $r$ between two language distance measures, we use the Representational Similarity Analysis (RSA) framework of Kriegeskorte et al. (2008); Chrupała and Alishahi (2019): we produce two matrices of all pairwise distances between 282 Wikipedia languages and we compute Pearson’s $r$ between the upper-triangulars, excluding the diagonals.

To make it easy for others to reproduce and build upon our work, we have published a reference implementation of our $n$-gram coverage model, which suggests subword sizes for fastText models. The reference implementation contains pre-computed subword frequencies for 288 Wikipedia languages, which makes the suggestions instantaneous.

---

*For six out of the 288 Wikipedia languages, Littell et al. did not provide feature vectors: Bhojpuri (bh), Emilian-Romagnol (eml), Western Armenian (hyw), Nahuatl (nah), Simple English (simple), and Sakizaya (szy).*

*See [https://github.com/MIR-MU/fasttext-optimizer](https://github.com/MIR-MU/fasttext-optimizer).*
Table 2: Accuracies on English, German, Czech, Italian, Spanish, French, Hindi, Turkish, and Russian word analogy tasks. Optimal subword sizes for the different word analogy tasks are **bold**: 4–5 for English, 6–6 for German, 1–4 for Czech, 2–5 for Italian, 3–5 for Spanish, 6–6 for French, 2–2 for Hindi, 3–3 for Turkish, and 5–6 for Russian. Our training and testing word analogy tasks are shown on separate lines.

<table>
<thead>
<tr>
<th></th>
<th>(a) English</th>
<th>(b) German</th>
<th>(c) Czech</th>
<th>(d) Italian</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>1 4 4 4 5</td>
<td>1 4 4 4 6</td>
</tr>
<tr>
<td>3</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>1 4 4 4 5</td>
<td>1 4 4 4 6</td>
</tr>
<tr>
<td>4</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>1 4 4 4 5</td>
<td>1 4 4 4 6</td>
</tr>
<tr>
<td>5</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>1 4 4 4 5</td>
<td>1 4 4 4 6</td>
</tr>
<tr>
<td>6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>1 4 4 4 5</td>
<td>1 4 4 4 6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(e) Spanish</th>
<th>(f) French</th>
<th>(g) Hindi</th>
<th>(h) Turkish</th>
<th>(i) Russian</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
</tr>
<tr>
<td>3</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
</tr>
<tr>
<td>4</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
</tr>
<tr>
<td>5</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
</tr>
<tr>
<td>6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
<td>2 3 4 5 6</td>
</tr>
</tbody>
</table>

Table 3: The n-gram coverages for English, German, Czech, Italian, Spanish, French, Hindi, Turkish, and Russian. The n-gram coverages for the optimal subword sizes on the different word analogy tasks are **bold**: 3.76% for English, 4.19% for German, 3.28% for Czech, 3.81% for Italian, 4.32% for Spanish, 9.20% for French, 0.57% for Hindi, 0.57% for Turkish, and 14.32% for Russian.
<table>
<thead>
<tr>
<th>Language</th>
<th>Default subword sizes</th>
<th>Suggested subword sizes</th>
<th>Optimal subword sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3–6</td>
<td>5–5</td>
<td></td>
</tr>
<tr>
<td>Spanish</td>
<td>57.00</td>
<td>57.60 (5–5)</td>
<td>57.60 (5–5)</td>
</tr>
<tr>
<td>French</td>
<td>68.38</td>
<td>69.33 (5–5)</td>
<td>69.60 (6–6)</td>
</tr>
<tr>
<td>Hindi</td>
<td>12.87</td>
<td>15.03 (1–3)</td>
<td>16.95 (2–2)</td>
</tr>
<tr>
<td>Turkish</td>
<td>38.04</td>
<td>38.34 (1–4)</td>
<td>39.51 (3–3)</td>
</tr>
<tr>
<td>Russian</td>
<td>51.89</td>
<td>52.51 (5–5)</td>
<td>52.75 (5–6)</td>
</tr>
</tbody>
</table>

Table 4: Accuracies on the Spanish, French, Hindi, Turkish, and Russian word analogy tasks using the default subword sizes of Bojanowski et al. (3–6) and Grave et al. (5–5), the subword sizes suggested by n-gram coverage, and the optimal subword sizes. Best accuracies for each language are **bold**, second best are in *italics.*

4 Results

In this section, we show and discuss the optimal subword sizes, accuracies, and n-gram coverages on the English, German, Czech, Italian, Spanish, French, Hindi, Turkish, and Russian word analogy tasks. We also show that the n-gram coverage can be used to suggest subword sizes that are close to the optimal subword sizes.

4.1 Optimal subword sizes

In Table 2 on the previous page, we show the accuracies and the optimal subword sizes on the English, German, Czech, Italian, Spanish, French, Hindi, Turkish, and Russian word analogy tasks. The optimal subword sizes 4–5 for English and 6–6 for German reproduce and confirm the results of Bojanowski et al. (2017, Section 5.2).

The optimal subword sizes for English (4–5), Italian (2–5), Spanish (5–5), French (6–6), and Russian (5–6) word analogy tasks are equal or within 1% accuracy of the default subword sizes suggested by Bojanowski et al. (3–6) and Grave et al. (5–5). In contrast, we see an improvement of up to 14% for Czech (1–4), 5% for Hindi (2–2), 4% for Turkish (3–3), and 3% for German (6–6).

To understand these differences, we look to the linguistic typology of languages: Czech, Hindi, and Turkish are synthetic languages and benefit from short subwords that represent morphemes. German and Russian are also synthetic, but the long compound nouns in German and the use of separate characters for yers (ъ and ь) in Russian make both languages benefit from longer subwords.

4.2 N-gram coverage

In Table 3 on the preceding page, we show the n-gram coverages for English, German, Czech, Italian, Spanish, French, Hindi, Turkish, and Russian. The mean n-gram coverage for the optimal subword sizes on the training word analogy tasks (English, German, Czech, and Italian), which we use to suggest subword sizes for the testing word analogy tasks (Spanish, French, Hindi, Turkish, and Russian), is 3.76%. The mean n-gram coverage for the optimal subword sizes on all word analogy tasks, which can be used in applications of fastText to suggest the optimal subword sizes, is 4.91%.

4.3 Suggested subword sizes

In Table 4, we compare the accuracies on the testing word analogy tasks (Spanish, French, Hindi, Turkish, and Russian) using the default subword sizes of Bojanowski et al. (3–6) and Grave et al. (5–5), the subword sizes suggested by the n-gram coverage, and the optimal subword sizes.

Using the suggested subword sizes is never worse than using the default subword sizes. For Hindi and Turkish, the suggested subword sizes always improve the accuracy: by 2.58% on average compared to the weaker default subword sizes and by 1.23% on average compared to the stronger default subword sizes. For Spanish, French, and Russian, the suggested subword sizes equal the default subword sizes of Grave et al. and they improve the accuracy by 0.72% on average compared to the suggested subword sizes.

For Spanish, the optimal subword sizes equal the suggested subword sizes. For French, Hindi, Turkish, and Russian, the optimal subword sizes improve the accuracy by only 0.90% on average compared to the suggested subword sizes, whereas they improve the accuracy by 2.59% on average compared to the weaker default subword sizes and by 1.52% on average compared to the stronger default subword sizes.
Subword sizes have a profound impact on the accuracy of fastText language models and their word embeddings. However, they are expensive to optimize on large corpora. In this work, we showed the optimal subword sizes for Czech, Italian, Spanish, French, Hindi, Turkish, and Russian fastText language models, we confirmed prior optimal subword sizes reported for English and German, and we showed that the optimization of subword sizes improves the accuracy of fastText on word analogy tasks by up to 14% compared to the default subword sizes. Our optimal subword sizes can be used in applications of fastText as the new default.

Furthermore, we proposed a cheap and simple n-gram coverage model that consistently improves the accuracy of fastText models on the word analogy tasks by up to 3% compared to the default subword sizes, and that it is within 1% accuracy of the optimal subword sizes on average. Subword sizes suggested by our n-gram coverage model can be used in applications of fastText as the new default for under-resourced languages, where the optimal subword sizes are unknown.

## 6 Future work

Although the word analogy intrinsic task is a convenient proxy for the usefulness of fastText word embeddings, Ghannay et al. (2016); Chiu et al. (2016); Rogers et al. (2018) show that it is no substitute for actual extrinsic end tasks. In future work, we will evaluate our n-gram coverage model on extrinsic tasks.

In recent machine translation models (Vaswani et al., 2017), text is tokenized into words and subwords using word-piece (Wu et al., 2016) and byte-pair (Sennrich et al., 2016) models. Our experiments suggest that we can remove the subword size parameter from fastText models and draw subwords from byte-pair models with little adverse effect on the word analogy accuracy. In future work, we will evaluate the use of word-piece and byte-pair models for subword selection in fastText models both on the intrinsic word analogy task and on other extrinsic tasks.

### References


Anna Rogers, Shashwath Hosur Ananthakrishna, and Anna Rumshisky. 2018. What’s in your embedding, and how it predicts task performance. In Proceedings of the 27th International Conference on Computational Linguistics, pages 2690–2703, Santa Fe, New Mexico, USA. ACL.


1078
CLexIS²: A New Corpus for Complex Word Identification Research in Computing Studies

Jenny A. Ortiz-Zambrano
Universidad de Guayaquil
Guayaquil, Ecuador
jenny.ortizz@ug.edu.ec

Arturo Montejo-Raéz
CEATIC - Universidad de Jaéén
Jaéén, Spain
amontejo@ujaen.es

Abstract

Reading is a complex process not only because of the words or sections that are difficult for the reader to understand. Complex word identification (CWI) is the task of detecting in the content of documents the words that are difficult or complex to understand by the people of a certain group. Annotated corpora for English learners are widely available, while they are less common for the Spanish language. In this article, we present CLexIS², a new corpus in Spanish to contribute to the advancement of research in the area of Lexical Simplification, specifically in the identification and prediction of complex words in computing studies. Several metrics used to evaluate the complexity of texts in Spanish were applied, such as LC, LDI, ILFW, SSR, SCI, ASL, CS. Furthermore, as a baseline of the primer, two experiments have been performed to predict the complexity of words: one using a supervised learning approach and the other using an unsupervised solution based on the frequency of words on a general corpus.

1 Introduction

Reading is a complex process not only because of the words or sections that are difficult for the reader to understand. Therefore, an adequate understanding of the content of the texts is required to be able to create coherent mental representations and this way to be able to capture their content (van den Broek, 2010).

Information technologies make it possible for people to access abundant information in different areas such as education, news, social, health, or government, among others. However, this information is not accessible to many, since some people face great reading barriers such as long sentences, unusual words, or complex linguistic structures that do not allow them to understand the content of the texts, with people with intellectual disabilities and people being directly affected in learning; including university students, who are people with a high educational level and specialized knowledge in different subjects of study but, still, could be part of groups of people with reading disabilities (Alarcón et al., 2020).

Complex Word Identification (CWI) is the task of detecting words in the contents of texts that are difficult or complex for people in a certain group to understand (Rico-Sulayes, 2020). CWI and the substitution of words identified as complex may significantly improve readability and understandability of a given text (Zotova et al., 2020).

In recent years, CWI has become an area of great interest for the scientific community and researchers in computational linguistics proposing development of computational semantic analysis systems as evidenced by the shared tasks of CWI by (Paetzold and Specia, 2016) in SemEval 2016, and NAACL-HTL 2018 by (Yimam et al., 2018), the task of the CWI of the ALexS 2020 contest, headquarters of IberLEF 2020 by (Ortiz-Zambranoa and Montejo-Ráezb, 2020), and 15th edition of SemEval but the first Lexical Complexity Prediction task. (Shardlow et al., 2021).

Annotated English Learner Corpus are widely available, Spanish Large Learner Corpus are far less common (Davidson et al., 2020). Although there are corpus for Natural Language Processing (NLP) research in Spanish, they do not contain the necessary annotations to develop reading comprehension tools for students in computing science.

Our aim is to begin to address the lack of data recorded in the corpora of written learner Spanish. This article introduces the creation of a new corpus in Spanish to contribute to the advances of research in the area of Lexical Simplification, specifically in the identification and prediction of complex words in computing studies.

The corpus is named CLexIS², and it is made
up of a collection of academic texts from the degrees in Computer Systems Engineering and the Software degree of the Faculty of Mathematical Sciences of the University of Guayaquil (Ecuador), a public institution, and one of the largest and oldest in the country, with around 67,000 students (according to the census in 2019).

2 Related work

2.1 Corpora for CWI in Spanish.
(Pitkowski and Gamarra, 2009) insure that a corpus is a large collection of different types of texts, oral or written, in electronic format, made up of tens of thousands of words, and in some cases made up of several million words. The processing of these large amounts of electronic texts contributes significantly to its application in numerous areas of study in the field of linguistics such as learning a second language (L2), lexical and syntactic simplification, predictions, automatic translations, retrieval and information extraction, speech synthesis, language analysis, among others. (Davidson et al., 2020) state that few corpus written in Spanish are available to NLP researchers. Some corpus for Spanish do not usually include annotations that facilitate the development of NLP models.

(Ortiz-Zambranoa and Montejo-Rázeb, 2020) recently created a resource that can be used to test complex difficult word identification systems, built to adapt to the educational environment. It is a new annotated corpus of transcripts of teaching classes, called VYTEDU-CW. This resource was provided for the ALeXs workshop (Task on Lexical Analysis at SEPLN 2020) as part of the second edition of IberLEF 2020 (Iberian Languages Evaluation Forum) that joined the efforts of the IberEval and TASS workshops where participants applied interesting approaches to address the CWI problem in an unsupervised or semi-supervised way.

(Davidson et al., 2020) generated the data corpus of Spanish students Corpus of Written Spanish of L2 and Heritage Speakers, or COWS-L2H built to help researchers better understand L2 development, examine practices teaching empirically and develop NLP tools and thus provide a better service for the community of Spanish teachers. This resource consists of 3.498 short essays written by students at an American university.

(Miaschi et al., 2020) presented an NLP-based approach to track the evolution of written language proficiency in L2 Spanish students using a wide range of linguistic characteristics automatically drawn from students’ written productions. To carry out their purpose, they analyzed the development of students’ writing from the COWS-L2H (Davidson et al., 2020).

The Complex Word Identification (CWI) Shared Task organized as part of the 13th Workshop on Innovative Use of NLP for Creating Educational Applications (BEA), hosted in conjunction with NAACL-HLT’2018, focused on multilingualism and provided data sets containing four languages: English, German, French, and Spanish. According to (Yimam et al., 2018) the goal of the CWI task was to predict which words challenge non-native speakers based on annotations collected from native and non-native speakers.

(Parodi, 2015) proposed the Corpus of Spanish Learners (CAES - acronym in Spanish) (Corpus of Spanish learners in English). (Segura-Bedmar and Martinez, 2017) used the EasyDPL (Easy Drug Package Leaflets) corpus, a collection of 306 booklets written in Spanish and manually annotated with 1400 adverse drug effects and their simplest synonyms. The objective of this work was to improve the readability of leaflets by replacing the terms that describe the effects of drugs with simpler synonyms. They used a vector from a previously trained word embedding model.

2.2 Lexical Complexity Measures.
A good indicator of writing quality is to use a measure of lexical complexity, referring to the size, variety, and quality of a student’s vocabulary (Crossley et al., 2012). The task of detecting in the content of the documents the words that are difficult or complex to the people of a certain group is known as complex word identification (CWI) (Rico-Sulayes, 2020). Replacing these words with their simplest synonym can improve the understandability and readability of a given text (Zotova et al., 2020). This process may be adapted for college students by making texts more readable (Alarcón et al., 2020).

(Schnur and Rubio, 2021) conducted a study that focused on the application of lexical complexity operationalized by three measures: lexical diversity, lexical density, and lexical sophistication using the 2.4 million-word written Spanish subsection of the Corpus of Utah Dual Language Immersion. The study investigated the effect of the three measures of lexical complexity where it was shown that a broad and deep lexical repertoire is a key charac-
teristic of the most advanced levels of proficiency.

In the research carried out by (Saggion et al., 2015) in the Implementation and Evaluation of a Text Simplification System for Spanish, they applied the Lexical Readability Measures based on the definitions of (Rebollo, 2008) for the calculation of low frequency words.

(Kajiwara and Komachi, 2018) introduced systems named TMU for the identification of complex words. TMU systems applied random forest classifiers and regressors whose characteristics were the number of characters and words and the frequency of target words in various corpus.

To characterize the corpus we applied in this work several metrics used to evaluate the complexity of texts in Spanish were applied, such as LC, LDI, ILFW, SSR, SCI, ASL, CS considered as an approach to validate the coherence of the manually annotated terms regarding their complexity.

3 A New Dataset

The creation of $CLexIS^2$ gave rise to a new data set in the scope of academic courses at a higher education level. The process of preparing the texts is detailed below:

As a first step, the subjects that make up each semester of study were identified. The first four semesters correspond to the Software career and the following four semesters to the Computer Systems Engineering career, giving a total of eight study semesters, where each semester consists of five subjects.

Next, the recordings of the classes (academic videos) taught by teachers in virtual classes in the last two semesters of study were selected, which were stored in each teacher’s work cloud.

Using the Dictation application, the transcription process of each of the academic videos was carried out. The automatic transcriber did not have precision in the accentuation of the words as well as in the punctuation of the sentences, therefore, a process of grammar revision was carried out manually in each text to achieve its correct accentuation; it was also considered vitally important to separate the text into sentences for better understanding.

Finally, the number of texts per subject corresponds to an average of 100 texts, and in turn, each text contains an average of 77.29 words. Table 1 shows the descriptive statistics of $CLexIS^2$, with a total of 3,887 texts. In Table 2 the definition of the variables is detailed.

3.1 Manual Annotation.

The students who participated in the labeling $CLexIS^2$ are ecuadorian university students with enrollment in the Computer Systems Engineering career, and the Software career in the regular academic period 2020-CII. Five annotators were chosen for each semester of study to carry out the corpus labeling work.

The average score of the academic performance of the participants according to their university expedient was 8.72 / 10 points. It should be noted that no distinctions were made in the selection of students who would carry out the process of labeling the complex words of the corpus texts. The students came from different levels of secondary education (private school, national - government), economic and geographical locations including vulnerable sectors such as suburban neighborhoods, rural parishes and several housing cooperatives located on the outskirts of the city.

3.2 Labeling Process.

An application was developed with free software tools Python, Fire-base, and Cloud Firestore for the creation and management of the database, and the texts were loaded into the system. The taggers had to begin to read the texts that corresponded to their study semester and then identify and write down the words that were difficult for them to understand.

The annotated data was collected for later management. The data set consisted of the following fields: the token (the difficult word), the annotator identification, the position of the token in the text, the name of the text, the length of the token and its frequency.

As can be seen in table 4, the columns in the table represent the number of scorers. Each row contains the semester of study and the total number of words rated as difficult in that semester. It is evident that the highest number of words labeled as complex is at the level of complexity of 0.2, they correspond to the total number of complex words identified by one annotator, which means that there are no coincidences that these words have been annotated by the other taggers.

A similar behavior occurs in the Computer Systems Engineering career, the total number of words annotated by a single tagger is much higher than when the words are annotated by more than one

---

1 Dictation - https://dictation.io/speech
4 Lexical Complexity of the Corpus

The evaluation of the complexity of the CLexIS² corpus texts was carried out by applying the seven measures of lexical complexity for the Spanish language as in (Saggion et al., 2015). These formulas were proposed by (Rebollo, 2008), with the exception of the SSR whose measurement was provided by (Spaulding, 1956). The detail is as follows:

The Lexical Complexity Index - LC.
Lexical Distribution Index - LDI.
Index of Low Frequency Words - ILFW.
Spaulding’s Spanish Readability Index - SSR.
The Sentence Complex Index - SCI.
The Average Sentences Length - ASL.
The Percentage of Complex Sentence - CS.

This evaluation was realized based on two factors, the first, at the lexical complexity of reading texts for which the readability indices LC, LDI, ILFW, and SSR were applied; and the second, the syntactic complexity of the texts where the measures applied were SCI, ASL, SC. For data processing, the open source statistical software Jasp version 0.14.1 was used, obtaining the descriptive statistics of the subjects that make up the Systems Engineering and Software degrees - 40 subjects in total. The table 3 shows the values that correspond to the lexical complexity metrics detailed in the previous paragraph.

The analysis of the results shows that the indices obtained determine that the texts corresponding to the first four semesters of the Software career and the remaining four to the Computer Systems Engineering career show an increase in terms of complexity at each semester, which represents that students who enter their university studies begin from the beginning to face the use and application of a new lexicon. As students are promoted to other semesters, the subjects to learn are new, and others correspond to the continuity of what was learned in the previous semester, which implies that students are constantly learning and using the technical vocabulary present in their studies.

The lexical complexity of the words per semester according to the results, determine that, in the case of the Systems Engineering career, the semesters of study correspond from the fifth semester to the eighth. The calculated index indicates a high complexity, with a value of 12,264.92, since in that semester the student learns subjects whose content involves a combination of programming languages and data that lead to the development of more complex solutions. These involved courses are: Database II, Organizational Behavior and Talent Human, Object Oriented Software Engineering, Artificial Intelligence, and Computational Organization and Architecture.

The LC has a slight decrease in the sixth semester down to 10,129.64; in that semester the course exhibits more theoretical subjects than practical, being these Elective III, Legislation in Computing, Financial Mathematics, Microprocessors,
### Table 3: Results by subject of the application of lexical complexity metrics.

<table>
<thead>
<tr>
<th>Software degree</th>
<th>Computer Systems degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of annotators</td>
<td>Number of annotators</td>
</tr>
<tr>
<td>Sem 1</td>
<td>2</td>
</tr>
<tr>
<td>1st</td>
<td>1615</td>
</tr>
<tr>
<td>2nd</td>
<td>1964</td>
</tr>
<tr>
<td>3rd</td>
<td>1930</td>
</tr>
<tr>
<td>4th</td>
<td>1316</td>
</tr>
</tbody>
</table>

Table 4: Results by semester of the careers of Software and Engineering in Computer Systems about the total number of complex words annotated by the taggers.

**Simulation.**

In the seventh semester, the LC shows an increasing order and reaches the value of 11,751.32; this is because the student has subjects whose LC is between 22301.81 and 26,971.89. The subjects are: Computer Center Administration, Compilers, Economics, Information Security and Distributed Operating Systems.

In the case of the eighth semester, the LC has a quite evident decrease, it decreases to 9,975.89, it is the last semester of studies for the student, it has subjects what has been learned is applied throughout their university stay, they are subjects that are more oriented to the administrative part and its approach is directed to the development of the student’s degree project, these subjects are: Systems Auditing, Elective IV, Finance, MIS (Information Systems Administration) and Management Information Systems.

### 5 The Experiments in CWI on the New Corpus

We carried out two experiments following two major approaches in CWI:

1. Detection of complex words based on the CREA resource. This is an unsupervised, lexicon based approach.

2. Prediction of complex words using a machine learning approach over different lexical features.

---

**Figure 1:** Lexical Complexity metrics in Computer Systems and Software Engineering careers by Semester of study.
detect if a word is complex or not based on CREA. Next, the confusion matrix was performed to determine the effectiveness of the classification system used.

We evaluated the effectiveness of this simple approach in terms of precision, recall, F1-score and accuracy. The results show that, the proportion of predictions that the model classified correctly corresponds to an accuracy of 0.4394, a precision of 0.5165 that belong to the really correct positive identifications, the hit rate allowed us to obtain a recall of 0.4709, and the proportion of real negatives correctly identified whose measure is the specificity with a value of 0.3963, finally, an $F1$ of 0.4927 was obtained that indicates the precision and robustness of the applied model.

The analysis of the data determines that the Software career has a higher Lexical Complexity than the Computer Systems career. Once again, first-semester students find it difficult to move from high school to undergraduate education. The subjects that first semester students have are: Programming Algorithms and Logic, Differential Calculus, Democracy, Introduction to Software Engineering, and Language and Communication.

5.2 Complex Word Identification using a Machine Learning Approach.

A supervised learning approach was applied using the Random Forest algorithm. The annotated data identifying the simple or multi-word complex word was necessary. Therefore, our system was developed following the process detailed below.

5.2.1 Training/Test Data.

Data from the annotated corpus CLexIS2-CW were used. We follow the example of the corpus data model provided by Lexical Complexity Prediction (LCP) shared task, organized by the International Workshop on Semantic Evaluation - SemEval-2021 (Shardlow et al., 2021) for Task 1: Lexical Complexity Prediction on the Lexical semantics track.

The data set consisted of the fields: Id of the text from which the complex word comes, the sentence, the word labeled as complex, and a level of complexity (computed as the division between the number of taggers who scored the word as complex and the total number of taggers). See Table 5.

5.2.2 Features.

To feed the learning algorithm, a total number of 15 characteristics were generated per sample, as in the works of (Gooding and Kochmar, 2018) y (Finnimore et al., 2019) for the detection of complex words:

- Absolute frequency (abs-frequency): the absolute frequency.
- Relative frequency (rel-frequency): the relative frequency of the target word.
- Word length (length): the number of characters of the token.
- Number of syllables (number-syllables): the number of syllables.
- Target word position (token-position): the position of the target word in the sentence.
- Number of words in the sentence (n-words-sentences): number of words in sentence.
- Relative frequency of the previous the token (freq-rel-word-before): the relative frequency of the word before the token.
- Relative frequency of the word after the token (freq-rel-word-after): the relative frequency of the word after the token.
- Length of previous word (len-word-before): the number of characters in the word before the token.
- Length of the after word (len-word-after): the number of characters in the word after the token.
- Measure of Textual Lexical Diversity (MTLD-diversity): the lexical diversity of the target word in the sentence using the metric proposed by (McCarthy and Jarvis, 2010), computed using this Python library.
- Number of synonyms (number-synonyms).
- Number of hyponyms (number-hyponyms).
- Number of hypernyms (number-hypernyms).
5.2.3 Applying Supervised Learning.

These numerical features were scaled to a standard range, because it has been proven that many machine learning algorithms when normalized are when they achieve the best results. Besides, a polynomial transformation with a degree value of 2 was applied to the characteristics which produced the creation of new characteristics. The Random Forest algorithm was selected as learning approach. To build the Random Forest regression model, the data set was divided into: the training set and the test set, where 10% of the data set was used as the test set and the remaining 90% was used as training set.

Several runs were performed with different configuration values to observe the performance of the algorithm and fine-tune the hyper-parameters of the model. The results on the test set lead the configuration with best scores, which consisted in 241 nodes and all the 15 features considered. The best results reported a MAE of 0.060970, MSE of 0.005889 and RMSE of 0.076739. See Table 6.

<table>
<thead>
<tr>
<th># Trees</th>
<th>MAE</th>
<th>MSE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>241</td>
<td>0.060970</td>
<td>0.005889</td>
<td>0.076739</td>
</tr>
<tr>
<td>241</td>
<td>0.060973</td>
<td>0.005888</td>
<td>0.076733</td>
</tr>
<tr>
<td>230</td>
<td>0.060980</td>
<td>0.005894</td>
<td>0.076771</td>
</tr>
</tbody>
</table>

Table 6: Best results obtained with the Random Forest algorithm.

6 Conclusions

A new corpus was created and made available. We believe that it becomes a fundamental resource for the identification of complex words in computer science studies, which means a very useful resource for the development of effective NLP tools for university students. The texts used as a central source of linguistic information reveal the difficulties faced by students of computer science studies.

The application of the lexical complexity metrics allowed evaluating the complexity of the content of the corpus texts, determining that in a large number of subjects, the lexicon that teachers use when teaching their classes contains complex sentences, a technical language and sophisticated causing difficulty in the understanding of students.

Future works could propose solutions that involve the creation of tools applying lexical simplification that greatly contribute to the contribution of students with low reading comprehension or intellectual disabilities to better understand the content of texts in the area of computer science. A possible solution will be the creation of a system that transforms complex texts into accessible ones, benefiting mainly university students in computer science who have disabilities and those who have reading comprehension difficulties.

The best result obtained for the predictive value of the words for the data set was: MAE of 0.060970, MSE of 0.005889 and RMSE of 0.09687 in the configuration with 240 nodes and 15 selected characteristics. The resource is available and can be shared by contacting the authors.

7 Acknowledgments

We appreciate Darwin Fabricio Borbor Merejildo, Kevin Francisco Labre Hidalgo, graduates of the Computer Systems Engineering degree from the University of Guayaquil, for their valuable contribution to the development of our work.

This study is partially funded by the Spanish Government under the LIVING-LANG project (RTI2018-094653-B-C21).
References


Towards Precise Lexicon Integration in Neural Machine Translation

Ogün Öz
Cobrainer GmbH
Munich, Germany
ogun.oz@cobrainer.com

Maria Sukhareva
Data Analytics Lab, Siemens AG
Nuremberg, Germany
maria.sukhareva@siemens.com

Abstract

Terminological consistency is an essential requirement for industrial translation. High-quality, hand-crafted terminologies contain entries in their nominal forms. Integrating such a terminology into machine translation is not a trivial task. The MT system must be able to disambiguate homographs on the source side and choose the correct wordform on the target side. In this work, we propose a simple but effective method for homograph disambiguation and a method of wordform selection by introducing multi-choice lexical constraints. We also propose a metric to measure the terminological consistency of the translation. Our results have a significant improvement over the current SOTA in terms of terminological consistency without any loss of the BLEU score. All the code used in this work will be published as open-source.

1 Motivation

The importance of consistent terminology has long been discussed by translation experts (Dagan and Church, 1994; Merkel, 1998; Itagaki et al., 2007; Saraireh, 2001; Byrne, 2006). Terminological standardisation is a critical task for technical and non-technical industrial translation. Patents, technical manuals, and medical instructions rely on consistent usage of technical terminology. But also non-technical news releases, marketing texts, promotion materials, legal and financial documents need to adhere to the same terminology. Byrne (2006) correctly points out that many large companies have their own terminologies that should be used in all texts. Such terminologies prescribe the correct usage of terms and provide not only a list of words that are to be used but also a list of their synonyms that should not be used by writers and translators (so-called negative terms). Sukhareva et al. (2020) describe such terminology for an automotive company and its usage in detail. Not adhering to these rules can be not only confusing for a reader but can also lead to serious legal and financial consequences if it is proven that damage was caused by the ambiguity of the instructions.

Morphologically rich languages also pose a very practical problem for terminology integration: terminological entries are provided in their nominative singular form (Susanto et al., 2020). The SOTA approaches rely on the assumption that the terminological entry can be found as is in the translated text. This is not the case for Slavic languages (e.g. Russian), for which finding the correct wordform on the target side is a key challenge for the terminology integration.

Morphologically poor languages (e.g. English), on the contrary, pose a very different challenge. Homographs appear in such languages not only due to polysemy and homonymy but also due to poor derivational morphology (e.g. a report vs. to report), thus, becoming a very common phenomenon. Liu et al. (2018) show that SOTA neural machine translation (NMT) fails to resolve homography efficiently. Despite being a known issue, the problem has received very little attention from the research community, and we are currently not aware of any prior work that would explicitly address the problem of homographs in the context of terminology integration into machine translation. This paper focuses on the following issues: resolving homographs when the source language is morphologically poor, choosing the right wordform in the morphologically rich target language, and evaluating terminological consistency in the resulting translation. We show that our approach for homograph disambiguation and morphologically flexible lexical constraints significantly improves terminological consistency as compared to the current SOTA.
2 Related Work

Previous work can be roughly divided into two groups: approaches that integrate lexicon during inference and approaches that integrate lexicon during training. A constrained decoding approach that has established itself as the SOTA in the past two years is Post and Vilar (2018). They proposed the Dynamic Beam Allocation (DBA) strategy, which decreased the decoding time complexity to constant time in respect to the number of lexical constraints. The proposed algorithm aims to allocate banks dynamically, prioritising the beams that satisfy the most constraints. This algorithm only allows incorporating a single wordform of a constraint, as Dinu et al. (2019) discussed in their work. This is a notable disadvantage of this approach as it assumes an unrealistic precondition that the provided lexical constraints will be correctly inflected. This condition cannot be satisfied when translating into a morphologically rich language.

On the contrary, training time approaches are more flexible in selecting the inflected forms. The SOTA in-training approaches tune a transformer model (Vaswani et al., 2017) towards producing translations that are biased towards an external lexicon. Song et al. (2019) proposed a simple way to copy target side terms into source sentences. Likewise, Dinu et al. (2019) suggested a source sentence modification method by replacing/appending target side terms using additional source factors. Nevertheless, these methods are only encouraging the model to use predefined target terms, whereas constrained decoding methods are enforcing terms’ usage. Thus, it can be argued that in-training approaches are inferior to the constrained decoding methods in terms of straightforward terminology integration and, indeed, Dinu et al. (2019) report the terminology usage rate 6-9% less than the constrained decoding method. To ensure the appearance of terms in the output, Michon et al. (2020) use placeholders with the help of morphosyntactic annotations. Even though the approach is effective for choosing a correctly inflected form, it depends on the availability and performance of morphological analysers both in source and target languages.

While all the aforementioned approaches have succeeded in improving the terminological consistency of translations, they essentially rely on a supervised selection of terminological entries. In other words, they assume that the homographs have already been resolved and a correct wordform is provided. Once the discussed approaches are set on a trial under realistic conditions, translation quality deteriorates. Word sense disambiguation is meanwhile a well-researched NLP task, and current state-of-the-art approaches can efficiently resolve homographs (Bohnet et al., 2018; Huang et al., 2019) but due to being time-consuming, are not applicable during translation inference.

3 Data

3.1 Parallel Corpus

For the training of the baseline NMT model, we used preprocessed bilingual WMT18 data\(^1\). We filtered out sentence pairs that have a length ratio of less than 1/3 or more than 3. We also applied language detection (\texttt{langid}) filtering (Lui and Baldwin, 2011) in a tolerant way: The sentence pairs for which \texttt{langid} could not predict the expected language in the first 10 predictions are filtered out. Finally, we removed 75,000 sentences with the worst alignment scores (Dyer et al., 2013). All the reported models utilize WordPiece (Wu et al., 2016) for tokenisation. To fine-tune the hyperparameters of the model, we used newstest2014, newstest2018, and newstest2019 as development sets. Newstest2017 is reserved for reporting the results. Since EN $\rightarrow$ RU newstest2020 was not available during the time of our experiments, we used RU $\rightarrow$ EN test set including an additional test set (test-ts\(^2\)), as a second set to report the results.

3.2 Terminology Extraction

Despite dictionaries of negative and positive synonyms being standard resources used by industrial translators, they usually cannot be openly shared. Thus, in order to ensure the reproducibility and comparability with previous work, we decided to use openly available resources: WMT Corpus and Russian Wordnet. We believe that such an approximation does not diminish the fairness of the evaluation as we are not focusing on domain adaptation but solely on improving lexical consistency of translation, which is just as applicable to and observable on news translations.

Tab.1 describes the process of generating our pseudo-dictionary of positive and negative terms. The Russian side of the training set is lemmatised

\(^1\)http://data.statmt.org/wmt18/translation-task/preprocessed/ru-en/
\(^2\)newstest2020-ruen-src-ts.ru and newstest2020-ruen-ref-ts.en
The boat’s engine had an emergency kill cord.

I opened it up to find out how the engine works.

<table>
<thead>
<tr>
<th>alignment</th>
<th>occurrence</th>
<th>in synset</th>
</tr>
</thead>
<tbody>
<tr>
<td>engine - двигатель</td>
<td>149</td>
<td>yes</td>
</tr>
<tr>
<td>engine - мотор</td>
<td>22</td>
<td>yes</td>
</tr>
<tr>
<td>engine - машина</td>
<td>4</td>
<td>no</td>
</tr>
<tr>
<td>engine - движущий</td>
<td>3</td>
<td>no</td>
</tr>
<tr>
<td>engine - механизм</td>
<td>3</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 1: The process of generating the terminological dictionary.

and matched against the Russian Wordnet (Chernobay, 2018). We use fast_align (Dyer et al., 2013) to extract word alignments of Russian and English sides of the training set. We proceed with finding the English word that is most frequently aligned to all the synonyms in a synset (e.g. "engine" is the most frequent match to "двигатель" dvigatel’ and "мотор" motor). This leaves us with a lexical entry for the English word "engine" and its Russian translations, which are the WordNet synonyms. Finally, we labelled the most frequently aligned Russian synonym in this list as a positive term, and all other Russian synonyms as negative terms (e.g. "двигатель" dvigatel’ is labelled as a positive synonym). Thus, from now on, if an English sentence has a word that occurs in our dictionary, the translator should resort to using the positive term in the translation and avoid negative terms. An example of a terminology entry can be found in Tab. 2.

3.3 Extraction of Wordforms

We further matched the terminology entries in the bilingual training data and kept track of the co-occurrence counts of inflected words to obtain a one-to-many list of wordform candidates per entry. Only the first candidate could be used as a lexical constraint for the related source phrase, whereas all the most frequent k options can be incorporated by our multi-choice lexical constraint approach. In order to extract Russian wordform candidates, we created a list of Russian wordforms most frequently aligned to a single inflected English wordform. As English is a morphologically poor language, we would end up with a list of Russian wordforms that would frequently contain five or more entries. Tab. 3 shows three distinct wordform lists of a terminology entry aligned to an inflected form of the English entry.

4 Approach

The approach consists of two major steps. On the source side of the morphologically poor language, it solves the problem of frequent homographs by applying a homograph disambiguator. On the target side of the morphologically rich language, it ensures that the translated term is correctly inflected.

4.1 Homograph Disambiguation for the Morphological Poor Language

Tab. 2 shows an entry in our terminology. All three Russian words are interchangeable synonyms in a certain context. But a straightforward string matching of word engine (Tab. 2) with an aim to force the translator to use a certain synonym in the target language would fail: the English word engine can also be used in the sense of a search engine (Fig. 1) which would have a Russian literal translation as "search system". In this case, the lexical constraint enforced by our terminology would not be correct

<table>
<thead>
<tr>
<th>Word</th>
<th>Lang</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>engine</td>
<td>en</td>
<td>Positive</td>
</tr>
<tr>
<td>двигатель</td>
<td>ru</td>
<td>Positive</td>
</tr>
<tr>
<td>мотор</td>
<td>ru</td>
<td>Negative</td>
</tr>
</tbody>
</table>

Table 2: Terminology entry

prevails: преобладает, преобладают, преобладающий
prevailing: преобладающих, преобладающие, преобладающее
prevailed: преобладал, преобладали, преобладало

Table 3: An example of extracted wordform options depends on the inflections in the source language.
and would cause poor translation quality.

To mitigate this problem, we propose a homograph disambiguation method. Our homograph disambiguation task is simpler than standard word-sense disambiguation (WSD) tasks (e.g. GlossBERT (Huang et al., 2019)) as it suffices to predict whether or not a certain word in the source sentence is used in the same sense as a terminology entry that has the same spelling and, unlike traditional WSD, there is no need to label all the possible senses of this word. We propose a word labelling model, similar to named entity recognition (NER) models, fine-tuned on BERT\footnote{BERT-Base, Cased (12-layer, 768-hidden)} (Devlin et al., 2019) having only two classes (\(T\) for Term and \(O\) for Non-Term). The model tags all the words in a sentence in one forward pass.

In order to create the training data for the homograph disambiguation, we used the same parallel corpus that we used for training the machine translation models. All the training data were processed with a word aligner \texttt{fast_align} (Dyer et al., 2013). All the sentences were lemmatised. Every lemma in the Russian sentence was compared against the extracted terminology (Sec. 3.2). If it is found in the terminology as a positive or negative term, we check whether the aligned English lemma is also listed as its translation (Tab. 2). If this is the case, the English word is labelled as "Term", otherwise as "Non-Term" (Fig. 1).

The BERT homograph tagger is fine-tuned for 4 epochs on this data.

### 4.2 Morphology Integration for the Morphologically Rich Language

As described in the Sec. 2, the Dynamic Beam Allocation (DBA)\footnote{For a detailed description of the DBA, refer to Post and Vilar (2018)} runs in constant time with respect to the number of constraints. The DBA accepts a list of constraint pairs (i.e. a term and its translation). During decoding, the candidates are grouped into banks with the number of banks equal to the number of constraints. If a term is found in the source sentence, then the translation candidates in which term’s translation occurs are propagated to a higher bank. The best translation is chosen from the bank with the highest rank (i.e. the ones that have the most satisfied constraints). The drawbacks of this approach is that it matches words without their context and can neither discriminate between homographs (addressed in the previous section) nor choose the correct inflection. As it forces a higher score on the translations that are compliant with the constraint list, the approach is not applicable to translating from a morphologically poor to a morphologically rich language as on one hand there are plenty of homographs on the source side and on the other hand there is a multitude of inflected wordforms on a target side. Constraining a translation on a wrong wordform (e.g., a nominative noun form instead of a dative form) would result in a translator giving a top score to a poor translation.

We propose multi-choice lexical constraints approach that overcomes DBA’s limitations and enables the translator to deal with morphologically rich languages by choosing a correct wordform. Similarly to (Post and Vilar, 2018), during inference we allocate candidates to banks. We find the longest possible (in terms of the number of tokens) candidate for every constraint to make sure there will be enough banks for all the possible constraints. Then to prioritise the entirely satisfied constraint phrases regardless of their token count, we rewarded them with the token count of the longest candidate. Without this change, the allocation strategy would be biased towards longer candidates.

**Number of constraints** The algorithm requires multiple banks to allocate candidate hypotheses. In the worst case, all the longest candidates would need a seat in the bank. For this reason, the number of constraints is the sum of the byte pair encoding (BPE) token counts of the longest constraint options. The size is calculated once since the constraint list remains unchanged during decoding. The number of constraints is calculated as follows:

\[
\text{size} = \sum_{c \in C} \max_o |c_o|
\]

where \(C\) is the constraint list, and \(o\) is a constraint \(c\) candidate in multi-choice lexical constraints (MLC) algorithm.

**Number of satisfied constraints** The satisfied constraint count of hypotheses decides in which bank they should be allocated. The number of banks equals to the maximum possible count if all the longest constraint variants are to be satisfied. However, as the algorithm is biased towards prioritising sentences with the most satisfied constraints, such sentences are longer and have higher overall cross-entropy loss. It causes a significant drop in the general quality of translations, especially if BPE tokenisation is used as more frequent
Figure 1: Labelling of the training data for homograph disambiguation: English words that are aligned to a synonym in Russian Wordnet synset are labelled as terms, otherwise they are considered to be homographs. “Двигатель” “dvigatel’” and “мотор” “motor” are found in the dictionary, while “система” “sistema” and “моторный” “motornyi” are not.

tokens are usually represented with fewer BPE tokens. To overcome this problem, we calculated the size of the satisfied constraints as follows: given $f(c)$ is the list of the advanced token indices of the constraint $c$’s variant, the number of satisfied constraints in a hypothesis is calculated as:

$$m(c) = \begin{cases} \max_{c_o} |c_o|, & \text{if } c \text{ is entirely met.} \\ \max_{c_o} f(c_o), & \text{if } c \text{ is advanced.} \\ 0, & \text{otherwise.} \end{cases}$$

$$\text{num\_met} = \sum_{c \in C} m(c)$$  \hspace{1cm} (2)

**Set of allowed constraints** We keep track of the advanced constraint to make sure we will advance on started but not entirely met constraints. However, when we have multiple variants for a constraint, even if the advanced constraint is known, we might have multiple variants of that constraint as advanced but not fulfilled yet. Therefore, we track the number of advanced tokens for all variants of the constraints. Finally, the set of allowed constraints is defined as the next tokens of all the advanced variants of the advanced constraint. If there is no advanced constraint, the set is simply the initial tokens of all the constraint options. The set $A(C)$ of all the allowed token indices is defined as:

$$A(C) = \begin{cases} f(c_o) + 1, & \exists c \text{ with advanced } o. \\ 0 \text{ for all } c, & \text{otherwise.} \end{cases}$$

(3)

**Advancing on constraints** The major difference to the DBA approach is that the advanced constraints have a list of variants on which the algorithm can advance in one step. Therefore, when there is an advanced constraint, all variants are considered as a possible advancement step. For instance, if the initial tokens of the constraint in example (1) are already advanced (поп, ##аж, ##ений ##ении) in decoding time step $i$, the algorithm advances on that constraint. The following tokens of both candidates are advanced together for the same hypothesis, which is a usual case when the choices have the same stem, and the only difference is the inflections. Its benefit is not only improving decoding run-time but also distributing the hypotheses more efficiently in the beams.

(1) поп, ##аж, ##ений ##ении

Fig. 2 shows that the run time of the MLC algorithm is comparable with the DBA (Post and Vilar, 2018) in different beam size settings and with different number of wordform choices.

Figure 2: Runtime comparison of (Post and Vilar, 2018) and multi-choice lexical constraints (MLC) as a function of wordform choices per constraints (average runtime per sentences with 2 constraint groups and similar sentence length) where $k$ is beam size.
The rest of the people will rest until the end of the year.

Table 4: An example sentence pair for terminology usage evaluation.

<table>
<thead>
<tr>
<th>src.</th>
<th>The rest of the people will rest until the end of the year.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tr.</td>
<td>Остальные люди будут отдыхать до конца года.</td>
</tr>
<tr>
<td>ref.</td>
<td>Остальные люди отдохнут до конца года.</td>
</tr>
</tbody>
</table>

5 Evaluation

All the models in our experiments were trained in the SOCKEYE toolkit (Hieber et al., 2017). The models that incorporate 6-layer, 8-head transformer architecture are trained 50 epochs on the training corpus (10,402,336 bilingual sentences after preprocessing). We modified the SOCKEYE toolkit to add the multi-choice lexical constraints algorithm and are going to publish the extension as an open-source.

For translation quality evaluation, we report BLEU score (Papineni et al., 2002) using SACREBLEU (Post, 2018), after detokenising the translations. Following Post and Vilar (2018), Dinu et al. (2019), and Susanto et al. (2020), we also report the terminology usage rate to evaluate terminological consistency.

5.1 Terminological F-score

Both BLEU score and terminological usage rate (Post and Vilar, 2018) are not sufficient to evaluate terminological consistency. The usage rate has proven to be seriously flawed as this metric does not account for homographs. Tab. 4 shows an example of a sentence translation that includes a homograph rest in its source sentence. Our terminology prescribes translating rest as a Russian adjective meaning “remaining” and does not contain an entry that would have the same meaning as its homograph verb to rest. The terminology usage rate used in the previous research was calculated in a rather straightforward manner by mere string matching. In our example, it would mean that the metric would only give a perfect score if the verb rest was incorrectly translated as its homograph adjective. If this were the case, despite the perfect score, the resulting translation would be of a very poor quality.

As Dougal and Lonsdale (2020) discuss, it is necessary to report an f-score metric when evaluating lexicon injected systems. Their suggested metric TREU intends to mitigate the negative effect of unmatched terminology tokens on BLEU metric assuming the reference sentences do not usually contain terminology promoted tokens. However, to assess the general quality of MT systems clearly, we find it more suitable to use the standard BLEU score. Thus, we require a separate metric based on the precision and recall of the terminology usage.

We propose a terminological f-score to account for precision and recall of the terminology usage in the hypotheses as compared to the reference translations. A similar metric was suggested to evaluate the performance of NMT models for the handling of homographs (Liu et al., 2018). The major difference between our metric and theirs is that we focus on the sense of the word rather than the string by consider all the aligned WordNet synonyms in the reference sentences. The precision and recall per sentence are calculated as follows:

$$
P = \sum_{l \in L_s} \min \left| l_s \right|, \left| l_f \right|, \left| l_r \right| \left| l_f \right|
$$

$$
R = \sum_{l \in L_s} \min \left| l_s \right|, \left| l_f \right|, \left| l_r \right| \min \left| l_s \right|, \left| l_r \right|
$$

where $L_s$ is the list of the terminological entries that occurred in the source sentence, $|l_s|$ is the occurrence number of terminology entry $l$ in the source sentence, $|l_f|$ is the occurrence number of the positive usage of that entry in the translation sentence, and $|l_r|$ is the occurrence number of both the positive and negative synonyms of the entry $l$ in the reference sentence. Thus, we calculate the precision and recall as 1/1 for the example in Tab. 4, whereas the terminology usage rate is 1/2.

5.2 Quantitative Results

Tab. 5 shows the results of the evaluation in terms of terminology usage rate, terminological f-scores, and BLEU scores for the newstest2017 and newstest2020 testsets. The baseline is a vanilla transformer model trained with the same parameters as all the other models without integrating the terminological dictionary. For the in-training baselines, we reproduce on our data the source-factoring (SF) model with append strategy that was described by Dinu et al. (2019). The inference time baseline is the lexical constraints (LC) approach by Post and Vilar (2018).
<table>
<thead>
<tr>
<th>Model</th>
<th>Term. Rate</th>
<th>Term. Prec.</th>
<th>Term. Recall</th>
<th>Term. F1</th>
<th>BLEU (Δ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>57.43</td>
<td>78.20</td>
<td>81.16</td>
<td>79.65</td>
<td>33.2</td>
</tr>
<tr>
<td>SF + BERT</td>
<td>81.22</td>
<td>62.76</td>
<td>95.54</td>
<td>75.76</td>
<td>30.2 (-3.0)</td>
</tr>
<tr>
<td>(Post and Vilar, 2018)</td>
<td>99.88</td>
<td>49.04</td>
<td>99.23</td>
<td>65.64</td>
<td>26.0 (-7.2)</td>
</tr>
<tr>
<td>MLC</td>
<td>99.68</td>
<td>50.82</td>
<td>99.54</td>
<td>67.29</td>
<td>28.2 (-5.0)</td>
</tr>
<tr>
<td>LC + BERT</td>
<td>61.67</td>
<td>75.02</td>
<td>87.30</td>
<td>80.69</td>
<td>31.1 (-2.1)</td>
</tr>
<tr>
<td>MLC random</td>
<td>70.71</td>
<td>66.92</td>
<td>86.55</td>
<td>75.48</td>
<td>31.7 (-1.5)</td>
</tr>
<tr>
<td>MLC + BERT</td>
<td>61.62</td>
<td>77.35</td>
<td>87.30</td>
<td>82.03</td>
<td>32.5 (-0.7)</td>
</tr>
</tbody>
</table>

(a) newstest2017

<table>
<thead>
<tr>
<th>Model</th>
<th>Term. Rate</th>
<th>Term. Prec.</th>
<th>Term. Recall</th>
<th>Term. F1</th>
<th>BLEU (Δ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>57.33</td>
<td>77.19</td>
<td>75.01</td>
<td>76.08</td>
<td>28.8</td>
</tr>
<tr>
<td>SF + BERT</td>
<td>81.42</td>
<td>64.72</td>
<td>92.72</td>
<td>76.23</td>
<td>26.4 (-2.4)</td>
</tr>
<tr>
<td>(Post and Vilar, 2018)</td>
<td>99.79</td>
<td>51.13</td>
<td>99.32</td>
<td>67.51</td>
<td>24.6 (-4.2)</td>
</tr>
<tr>
<td>MLC</td>
<td>99.51</td>
<td>52.46</td>
<td>99.15</td>
<td>68.62</td>
<td>24.9 (-3.9)</td>
</tr>
<tr>
<td>LC + BERT</td>
<td>63.90</td>
<td>74.35</td>
<td>84.73</td>
<td>79.20</td>
<td>27.4 (-1.4)</td>
</tr>
<tr>
<td>MLC random</td>
<td>72.31</td>
<td>65.17</td>
<td>82.54</td>
<td>72.83</td>
<td>27.3 (-1.5)</td>
</tr>
<tr>
<td>MLC + BERT</td>
<td>63.84</td>
<td>75.84</td>
<td>84.52</td>
<td>79.94</td>
<td>28.1 (-0.7)</td>
</tr>
</tbody>
</table>

(b) newstest2020 (extracted from ru-en wmt20/test-ts)

Table 5: Terminology usage and BLEU scores of baseline, source factoring by append (SF), lexical constraints (LC) and multi-choice lexical constraints (MLC) (ours) models.

Vilar (2018). We compare the baselines with the following proposed contributions:

1. Introducing homograph disambiguation (+BERT) as described in Sec. 4.1
2. Introducing multi-choice lexical constraints (MLC) for the inference approach as described in Sec. 4.2
3. Combining multi-choice lexical constraints and homograph disambiguation (MLC+BERT)

The evaluation shows that previously proposed SOTA methods for lexica integration by Dinu et al. (2019) and Post and Vilar (2018) suffer from a large decrease in the BLEU score. It also shows that the term usage rate used in the previous research is essentially meaningless for measuring translation quality as even though it has a nearly perfect score for Post and Vilar (2018), the BLEU score greatly dropped. Our approach, on the contrary, showed a significant improvement over all the baselines in terms of terminological f-score without decreasing translation quality. The reasons for the slight decrease of the BLEU score for MLC+BERT are discussed in detail in Sec. 5.3.

5.3 Qualitative Analysis

For a better insight into the results, we manually inspected the Russian translations. One of the primary reasons why MLC+BERT had a slight drop in the BLEU score as compared to the vanilla baseline was that the WMT testset was not tailored to have consistent terminology. We are also not aware of any open-source MT evaluation dataset with terminological consistency in mind. The evaluation showed that this was the reason for the drop in BLEU. Tab. 6 shows translations for which the BLEU score is lower for the MLC+BERT model. This hypothesis was tested by calculating the BLEU score for a subset of test sentences that contain the positive term in the Russian reference translation (80% of newstest2017 and 85% of newstest2017). The results in showed that the difference in the BLEU score between the baseline and our model decreases by more than double if all the test sentences with negative terms are eliminated (see Appendix A).

As compared to other baselines, our method greatly improves the quality of the translation for Post and Vilar (2018) and Dinu et al. (2019). Post and Vilar (2018) baseline is particularly prone to hallucinate Lee et al. (2018) if a lexical constraint
Table 6: An example from the newstest2020 evaluation set. The Russian gold sentence and the baseline contain a negative term. The MLC+BERT translation uses a positive interchangeable syllable. Even though the translation is perfectly fine, the BLEU score is lower for MLC+BERT.

<table>
<thead>
<tr>
<th>EN</th>
<th>Kvyat parked his <strong>car</strong> in one of the safety zones.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RU baseline</td>
<td>Квят припарковал свою <strong>машина</strong> в одной из зон безопасности.</td>
</tr>
<tr>
<td>MLC+BERT</td>
<td>Квят припарковал свой <strong>автомобиль</strong> в одной из зон безопасности.</td>
</tr>
</tbody>
</table>

is a homograph or is not correctly inflected (see Appendix B). In this case, the model generates an output till it reaches the maximum length. For example, the output of the LC baseline has 8% more characters than the reference translations. In comparison, the vanilla baseline has only 0.5% more characters and the MLC+BERT has exactly the same amount of characters. The manual evaluation showed that reducing hallucinations is the reason for the large increase of the BLEU as compared to the SF and LC baselines.

We also examined the effect of automatically generated lexicon on the translation quality. While we found cases in which positive terms were not perfect synonyms and were not interchangeable with negative terms, the homograph disambiguation seemed to show certain robustness by labelling the English term only if they occurred in the context that was common for negative and positive Russian translations. While we still believe that better results could be achieved in real-life settings where a high-quality dictionary would be used, our examination showed that there was no unreasonable error propagation from the usage of an automatically extracted dictionary.

The greatest weakness that we found during qualitative examination lies in how the top inflected candidates are scored in MLC. The MLC model takes a list of top \(n\) Russian wordforms that are most frequently aligned to a given English wordform of a term. In rare cases, an acceptable wordform does not appear to be in the top \(n\) list. In this case, the translation ends up being grammatically incorrect or hallucinates in a similar sense as the LC baseline.

A possible solution for this would be generating the top \(n\) choices for MLC in a more elaborated manner e.g. by considering the position in the sentence or even using syntactic information. For now, we leave exploring those options for future work.

### 5.4 Evaluation of Homograph Disambiguation

The homograph disambiguator was trained on artificially created labels, and we are not in possession of any gold standard data for the direct evaluation. We assume that evaluating the approach on the artificially labelled data will not ensure the objectivity of such an evaluation and both train and testset will contain the same errors. For transparency, we still provide the scores in Appendix (Tab. 8).

Thus, measuring the effect of homograph disambiguator on the downstream translation task is more sound. To make sure that the improvement of the terminological f-score is caused by the homograph disambiguation and not by the reduction of the number of lexical constraints, we introduce the MLC\textsubscript{random} baseline (see Tab. 5). We have calculated the total amount of constrained terms after applying the homograph disambiguation (+BERT) and randomly labelled the same amount of terms to be constrained in the original testsets. The evaluation results showed that the f-score dropped by 7% for the randomly labelled dataset, thus, proving that our homograph disambiguation is the actual cause of the f-score’s improvement.

### 5.5 Runtime Analysis

In order to ensure that MLC is also feasible for real-life usage, we compared the inference speed between the Post and Vilar (2018) and our MLC input (Fig. 2). As well as the DBA algorithm, MLC makes sure that the number of hypotheses is limited by the beam size. Thus, the runtime complexity of our approach is constant in the number of constraints. We have made an interesting observation that MLC is actually faster than LC for the beam size of 5 and slightly slows down for the beam size of 10. We have found the following explanation for such behaviour: Lexical constraints expect a large beam size in order to be able to generate enough hypotheses with the provided lexical constraints. The DBA does not allow a beam to generate the end of sentence symbol unless the constraints are met. Once a translation is incorrectly constrained on a homograph or on a wordform that cannot occur in translation, the beam cannot terminate unless it reaches the maximum length, and, thus, it negatively influences the inference time. On the con-
trary, the MLC allows a beam to terminate which makes it more time efficient.

6 Conclusion

We have presented an approach for terminology integration into a neural machine translation from a morphologically poor into a morphologically rich language. Our work makes the following contributions:

1. Disambiguation of the homographs in the morphologically poor language.

2. Multi-choice lexical constraints to ensure the correct choice of an inflected target wordform in the morphologically rich language.

3. A metric that takes into account precision and recall of terminology usage.

We propose a solution to the problem of rich morphology in the target language by presenting multi-choice lexical constraints and show that our combined approach (MLC+BERT) has a significantly\(^8\) better f-score than all the other models.

References


\(^8\)We used McNemar’s significance test (McNemar, 1947). The significant difference is defined as \(p < 0.05\).


A Further Quantitative Analysis

A.1 BLEU Scores of Filtered Datasets

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU (Δ)</th>
<th>Model</th>
<th>BLEU (Δ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>33.7</td>
<td>baseline</td>
<td>29.6</td>
</tr>
<tr>
<td>SF + BERT</td>
<td>32.3 (-1.4)</td>
<td>SF + BERT</td>
<td>29.3 (-0.5)</td>
</tr>
<tr>
<td>MLC + BERT</td>
<td>33.2 (-0.5)</td>
<td>MLC + BERT</td>
<td>29.3 (-0.3)</td>
</tr>
</tbody>
</table>

(a) newstest2017  
(b) newstest2020

Table 7: BLEU scores after filtering the sentences having at least one negative term.

A.2 Direct Evaluation of Homograph Disambiguation

<table>
<thead>
<tr>
<th>testset</th>
<th>precision</th>
<th>recall</th>
<th>f-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>newstest2017</td>
<td>76.17</td>
<td>56.64</td>
<td>64.97</td>
</tr>
<tr>
<td>newstest2020</td>
<td>66.92</td>
<td>79.90</td>
<td>72.84</td>
</tr>
</tbody>
</table>

Table 8: Evaluation table for homograph disambiguation task. Since there is no gold labels, predicted labels are compared against the artificially created labels.

B Qualitative Comparison of Translation Systems

| Type of error | Hallucination after correct translation  
|---------------|----------------------------------------|
| EN            | It was earlier reported that capital CSKA player Konstantin Kuchaev spoke out against the introduction of VAR (Video Assistant Referee).  
| RU            | Ранее сообщалось, что футболист столичного ЦСКА Константин Кучаев высказался против внедрения VAR.  
| baseline      | Ранее сообщалось, что столичный игрок ЦСКА Константин Кучаев выступил против введения ВАР (видео помощника судьи).  
| LC            | Ранее сообщалось, что столичный игрок ЦСКА Константин Кучаев выступил против введения ВАР (видео помощника судьи, который говорил о том, что арбитру докладе не удалось выступить с рефери).  
| MLC+BERT      | Ранее сообщалось, что столичный игрок ЦСКА Константин Кучаев выступил против введения ВАР (видео помощника судьи).  
| Comment       | The LC model generates a string after comma (marked in italics) that does not occur in the source text nor meaningful in the context. It happens because the lexicon prescribes to translate "report" as a noun meaning "an account given of a particular matter" доклад, while the source actually has a homograph verb "to report". The LC model generates a correct translation and proceeds to hallucinate till it finally produces a sentence with "a report". It leads to not only longer nonsensical output but also to longer inference time. The homograph disambiguation (MLC + BERT) correctly marks "report" as a non-term, thus, preventing the model to force a constraint on this sentence  

Type of error | Hallucination with a grammatically correct sentence  
|---------------|--------------------------------------------------|
| EN            | As reported by Chempionat, the 41-year-old specialist flew into Moscow to weigh up the possibility of working at one of Russia’s clubs.  
| RU            | Как сообщает “Чемпионат”, 41-летний специалист прилетел в Москву, чтобы изучить возможность найти работу в каком-нибудь российском клубе.  
| LC            | Как сообщает Chempionat, 41-старый специалист вылетел в Москву, чтобы в докладе проанализировать возможность работы в одном из российских клубов.  
| MLC+BERT      | Как сообщает Chempionat, 41-летний специалист вылетел в Москву, чтобы изучить возможность работы в одном из российских клубов.  
| Comment       | As in the previous example, the LC model forces to use the homograph noun "a report" to be a translation of the verb "to report". Unlike the example above, the model does not produce a correct translation at any point and generates a sentence with an entirely different meaning: “As reported by Chempionat, the 41-year old specialist got on a flight to Moscow to analyse in his report possibilities of working at one of Russia's clubs.” This kind of translations are particularly dangerous, as it would be extremely difficult for a native speaker without looking at the source to detect that the translation completely fails to convey the meaning. The homograph disambiguation solves this problem and the translation is correct.  

Type of error | Hallucination with an ungrammatical sentence  
|---------------|--------------------------------------------------|
| EN            | Documents obtained by the publication, reveal that the owners of TikTok (ByteDance company) with the help of their app are promoting Chinese foreign policy goals overseas.  

1098
В документах, оказавшихся у издания, рассказывается, что владелец TikTok (компания ByteDance) с помощью приложения продвигает цели внешней политики Китая за рубежом. Документы, полученные публикацией, показывают, что владельцы TikTok (компании ByteDance) с помощью своего приложения продвигают китайские внешнеполитические цели за рубежом. Документы, полученные изданием, показывают, что владельцы TikTok (компании ByteDance) с помощью своего приложения продвигают китайские цели внешней политики за рубежом. Документы, полученные изданием, показывают, что владельцы TikTok (компании ByteDance) с помощью своего приложения продвигают китайские цели внешней политики за рубежом. Документы, полученные изданием, показывают, что владельцы TikTok (компании ByteDance) с помощью своего приложения продвигают китайские цели внешней политики за рубежом. Роман Зарипов, основатель нашего цифрового агентства, согласился с Богдановым: “Основные правила для пользователей TikTok перечислены в пользовательском соглашении: нельзя выкладывать шокирующий контент, дискриминационные высказывания и так далее”.

<table>
<thead>
<tr>
<th>Type of error</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>A wrong wordform as lexical constraint</td>
<td>Documents obtained by the publication, reveal that the owners of TikTok (ByteDance company) with the help of their app are promoting Chinese foreign policy goals overseas.</td>
</tr>
<tr>
<td>Inconsistent terminology usage in the test set</td>
<td>Roman Zaripov, founder of the Our Digital agency, agreed with Bogdanov: “The main rules for TikTok users are listed in the user agreement: no posting shocking content, discriminatory rhetoric and so on.”</td>
</tr>
<tr>
<td>Insufficient coverage by the lexicon</td>
<td>This historic trajectory cannot be stopped by anyone or any force, said Xiaoguang.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of error</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>The baseline forces to translate foreign as иностраный which is not applicable in this context. The LC baseline generates a nonsensical sequence of words. This type of error is less harmful that the one described above as a native speaker can immediately spot that translation is incorrect. The MLC+BERT solves this problem and the translation is correct.</td>
<td></td>
</tr>
<tr>
<td>Inconsistency in the test set</td>
<td>Both baseline and MLC+BERT produced correct translations. Word “main” is prescribed to be translated as “owners” as singular nominative “owner”.</td>
</tr>
<tr>
<td>Insufficient coverage by the lexicon</td>
<td>This historic trajectory cannot be stopped by anyone or any force, said Xiaoguang.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of error</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>The error described in the previous example was resolved by the homograph disambiguation. However, the LC + BERT model produced a grammatically incorrect translation as the constraint for word “owners” was given in a wrongly inflected form of Genitive plural владельцев. The MLC+BERT solves this problem by providing a list of inflected forms and the result is a correct translation of the word in Nominative plural.</td>
<td>Both baseline and MLC+BERT produced correct translations. Word “main” is prescribed to be translated as foreign.</td>
</tr>
<tr>
<td>Insufficient coverage by the lexicon</td>
<td>This historic trajectory cannot be stopped by anyone or any force, said Xiaoguang.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of error</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>The LC baseline forces to translate foreign as иностраный which is not applicable in this context. The LC baseline generates a nonsensical sequence of words. This type of error is less harmful that the one described above as a native speaker can immediately spot that translation is incorrect. The MLC+BERT solves this problem and the translation is correct.</td>
<td></td>
</tr>
<tr>
<td>Insufficient coverage by the lexicon</td>
<td>This historic trajectory cannot be stopped by anyone or any force, said Xiaoguang.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of error</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>The error described in the previous example was resolved by the homograph disambiguation. However, the LC + BERT model produced a grammatically incorrect translation as the constraint for word “owners” was given in a wrongly inflected form of Genitive plural владельцев. The MLC+BERT solves this problem by providing a list of inflected forms and the result is a correct translation of the word in Nominative plural.</td>
<td>Both baseline and MLC+BERT produced correct translations. Word “main” is prescribed to be translated as foreign.</td>
</tr>
<tr>
<td>Insufficient coverage by the lexicon</td>
<td>This historic trajectory cannot be stopped by anyone or any force, said Xiaoguang.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of error</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>The error described in the previous example was resolved by the homograph disambiguation. However, the LC + BERT model produced a grammatically incorrect translation as the constraint for word “owners” was given in a wrongly inflected form of Genitive plural владельцев. The MLC+BERT solves this problem by providing a list of inflected forms and the result is a correct translation of the word in Nominative plural.</td>
<td>Both baseline and MLC+BERT produced correct translations. Word “main” is prescribed to be translated as foreign.</td>
</tr>
<tr>
<td>Insufficient coverage by the lexicon</td>
<td>This historic trajectory cannot be stopped by anyone or any force, said Xiaoguang.</td>
</tr>
</tbody>
</table>

Table 9: Examples of various errors that were identified during qualitative analysis
OffendES: A New Corpus in Spanish for Offensive Language Research

Flor Miriam Plaza-del-Arco, Arturo Montejo-Ráez, L. Alfonso Ureña-López and María-Teresa Martín-Valdivia

Department of Computer Science, Advanced Studies Center in ICT (CEATIC) Universidad de Jaén, Campus Las Llagunillas, 23071, Jaén, Spain
{fmplaza, amontejo, laurena, maite}@ujaen.es

Abstract

Offensive language detection and analysis has become a major area of research in Natural Language Processing. The freedom of participation in social media has exposed online users to posts designed to denigrate, insult or hurt them according to gender, race, religion, ideology, or other personal characteristics. Focusing on young influencers from the well-known social platforms of Twitter, Instagram, and YouTube, we have collected a corpus composed of 47,128 Spanish comments manually labeled on offensive pre-defined categories. A subset of the corpus attaches a degree of confidence to each label, so both multi-class classification and multi-output regression studies are possible. In this paper, we introduce the corpus, discuss its building process, novelties, and some preliminary experiments with it to serve as a baseline for the research community.

1 Introduction

Offensive language is defined as the text which uses hurtful, derogatory, or obscene terms made by one person to another person (Wiegand et al., 2019). Related terms in the literature are hate speech (Waseem and Hovy, 2016), cyberbullying (Rosa et al., 2019), toxic language (van Aken et al., 2018), aggression language (Kumar et al., 2018), or abusive language (Nobata et al., 2016). Although there are subtle differences in meaning, they are all compatible with the above general definition.

Due to the well-acknowledged rise in digital social interactions, in particular on social media platforms, the amount of offensive language is also steadily growing. Unfortunately, this type of prejudiced communication can be extremely harmful and could lead to negative psychological effects among online users, especially among young people, causing anxiety, harassment, and even suicide in extreme cases (Hinduja and Patchin, 2010).

At the same time, this issue also implicates governments, online communities, and social media platforms. In order to help fight this problem, these stakeholders are continuously taking appropriate actions to implement laws and policies combating hate speech. For instance, since 2013 the Council of Europe has sponsored the "No Hate Speech" movement1 seeking to mobilize young people to combat hate speech and promote human rights online. In May 2016, the European Commission reached an agreement with Facebook, Microsoft, Twitter, and YouTube to create a “Code of conduct on countering illegal hate speech online”2. From 2018 to 2020, platforms such as Instagram, Snapchat, and TikTok adopted the Code. According to a Spanish report in 2019 on the evolution of hate crimes in Spain3, threats, insults, and discrimination are counted as the most repeated criminal acts, with the Internet (54.9%) and social media (17.2%) as the most widely used media to commit these actions.

To help achieve this goal, automatic systems based on Natural Language Processing (NLP) techniques are required. To train these systems, corpora labeled on offensive language are essential. In recent years, the NLP community has invested considerable effort into resource generation. However, most of them have been directed towards English, even though it is a global concern and there are important cultural differences depending on the language examined. In addition, most of them have been focused on Twitter data, despite the presence of offensive language on other platforms such as YouTube or Instagram, which more widely used by young people.

To contribute to filling this gap, in this paper4

2. https://cutt.ly/Hj5EsAh
3. https://cutt.ly/ej5EgU7
4. NOTE: This paper contains examples of potentially ex-
we present OffendES, a Spanish collection of comments manually labeled for offensive content using a fine-grained annotation scheme. We collect our data from young influencers from well-known social platforms including Twitter, Instagram, and YouTube. Therefore, a comparative study of offensive behavior in social media and its relationship with the influencers is conducted. Finally, we propose preliminary experiments to serve as a baseline for the NLP community in which we show the validity of the corpus.

The remaining of the paper is organized as follows. Section 2 describes the related work on offensive language including some available datasets. Section 3 introduces our OffendES dataset and some descriptive statistics. Section 4 depicts our baseline evaluation of the novel dataset. A discussion is provided in Section 5. Finally, we conclude with our future studies in Section 6.

2 Related Work

2.1 Offensive Language Detection

In recent years, while offensive language continues to spread on the Internet, the importance of identifying this type of content in textual information has become increasingly significant in the NLP field, with several studies applying different machine learning systems. Most of these studies focus on the detection of offensiveness in social media, usually including a binary classification task to detect the presence of offensive language in the text.

Early studies explored traditional machine learning algorithms including Support Vector Machines, Logistic Regression, Random Forest, or Decision Trees, as well as the combination of different types of syntactic, lexical, semantic, and sentiment features (Chen et al., 2012; Nobata et al., 2016; Oråsan, 2018; Plaza-del-Arco et al., 2019).

As neural network architectures have shown promising results, extensive studies have recently explored a variety of deep learning architectures including Recurrent and Convolutional Neural Networks (Ransinghe et al., 2019; Sharifirad and Matwin, 2019; Georgakopoulos et al., 2018). More recently, Transformer-based models have made significant progress and represent the state-of-the-art of multiple tasks, including offensive language detection (Plaza-del-Arco, Flor Miriam and Molina-González, M. Dolores and Ureña-López, L. Alpícit or offensive content which may be offensive to some readers. They do not represent the views of the authors.

2.2 Data Available

Several labeled datasets are publicly available and usually include a binary annotation, indicating whether the content is offensive or not. Most of them have been generated in the context of different shared tasks for different languages.

For instance, the well-known offensive language task OffensEval has held two editions in the International Workshop on Semantic Evaluation (SemEval). In the first edition, Zampieri et al. (2019b) released the OLID dataset which contains over 14,000 English tweets. It was annotated using a three-level hierarchical annotation model by two people using a crowd-sourcing platform (Zampieri et al., 2019a). In order to retrieve tweets, they selected specific keywords and constructions often included in offensive posts related to Twitter accounts. Following the same annotation scheme, in the second edition Zampieri et al. (2020) introduced multilingual datasets comprising five different languages.

The Germeval shared task focused on offensive language identification in German tweets (Wiegand and Siegel, 2018). A dataset of over 8,500 annotated tweets was provided following also a hierarchical annotation. To collect the data, the authors explored the timeline of users that regularly post offensive content. Tweets were manually annotated by one of the three organizers of the task, and to measure inter-annotation agreement, 300 tweets were annotated by the three annotators in parallel. The annotation scheme is similar to the previously shared task, but differs in the following aspects: the number of levels in the hierarchy, the labels in the second level, and the language.

Related to Spanish, most of the datasets within the context of offensive language target hate speech, including AMI (Fersini et al., 2018), HateEval (Basile et al., 2019), and the HaterNet (Pereira-Kohatsu et al., 2019) collections. However, there is a lack of resources regarding the Spanish offensive language. To the best of our knowledge, the first corpus appeared at the 3rd SEPLN Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval) (Carmona et al., 2018). This corpus was also used in the next edition of this workshop in 2019 (Aragón et al., 2019). The dataset focuses on the Mexican variant of Spanish
and contains around 10,475 tweets binary labeled as offensive or non-offensive. This collection has been recently revised (Díaz-Torres et al., 2020). EmoEvent (Plaza-del-Arco, Flor Miriam and Strapparava, Carlo and Ureña López, L. Alfonso and Martín-Valdivia, María-Teresa, 2020) is a multilingual emotion corpus based on different events, it also includes a small proportion of tweets labeled as offensive. Finally, the DETOXIS task recently introduced the first dataset of comments in response to news articles labeled at different toxicity levels. To the best of our knowledge, there is no other Spanish corpus available with fine-grained categories for offensive language focused on young people. As the authors point out in (Aragón et al., 2019), the characterization of the offensiveness level found in a text is complex; therefore, there is a need for a more detailed classification of the tweets.

Our dataset, OffendES, differs from existing Spanish offensive language datasets because (i) apart from Twitter, we study the problem of offensive language detection on YouTube and Instagram, platforms that young people are more used to, (ii) we collect the data with a focus on young influencers, and (iii) we propose an annotation scheme with fine-grained classification.

3 OffendES Dataset

In this section, we describe the context of the dataset, the methodology followed to collect it and the annotation scheme proposed to label offensive content. Besides, we give some descriptive statistics and a detailed analysis of the collected data. OffendES is available upon request to the authors.

3.1 Scope of the Dataset

To understand the rationale behind the design and generation of the corpus, certain contextual information may be useful. As stated in the introduction, dealing with offensive posts in social networks is a growing concern. Several platforms are clear on this issue, as can be read in rules and policies of Twitter⁶, Instagram⁷ or YouTube⁸. Indeed, YouTube has disabled comments on videos and channels featuring children (The YouTube Team, 2019). But this is a major concern not only for platform providers but for public administrations, in order to limit the possible side effects of harmful messaging to more vulnerable communities, like children or teenagers. With this in mind, the creation of this resource aims to achieve the following long-term goals:

1. Early detection of offensive language use in social media on the Internet, with a special focus on young people.
2. Identifying improvements in protection systems for young people in social networks.
4. Creating a reference corpus for the study of language technologies applied to the classification of sexist language.

3.2 Data Collection

Instagram, YouTube, and Twitter are among the social media platforms most used by people ages from 18 to 24 (Jenn Chen, 2020). These three have been selected as the main data sources. A total of 12 controversial influencers with a significant number of followers have been identified and their respective accounts in the three selected media. Table 2 (Appendix) shows the accounts used by the selected influencers in the three selected media. They are Spanish influencers from 24 to 35 years old and, six are men and six are women. The process for collecting comments consisted of two main steps. To collect the data, first, the last 50 posts by each influencer were obtained using the platform API. Then, an ad hoc web scraper was launched to extract user comments to each of the posts obtained (limited to 2,000 replies). This script uses scrolling through JavaScript code commands to retrieve further comments. In the case of YouTube, instead of the scraper, its API⁹ has been used to retrieve comments.

During two months (from February to March 2020), a total number of 283,622 comments were collected (see Table 1 for detailed information). The comments were then filtered according to two main constraints: the presence of potentially offensive language and lexical diversity.

⁶https://cutt.ly/RkrVTQn
⁷https://cutt.ly/1j5Eut0
⁸https://cutt.ly/yj5Eijc
⁹https://cutt.ly/JkrVSYv
To avoid the creation of a corpus with few or no offensive comments set, we labeled all the comments with flags determining whether the comment contained any of the words found in five different controlled lexicons (Plaza-del-Arco, Flor-Miriam and Molina-González, M Dolores and Ureña-López, L Alfonso and Martín-Valdivia, M. Teresa, 2020). All comments with potentially offensive language were selected (23,788 comments). We selected 60,000 comments to be labeled in the manual annotation phase. Therefore, we selected 36,212 comments without offensive terms. Applying lexical diversity measures proved to be an interesting approach to ensure a diverse set of comments. Therefore, we first attempted to include those comments that added the highest lexical diversity value to the growing set of collected comments. To that end, we applied the Measure of Lexical Textual Diversity MTLD (McCarthy and Jarvis, 2010), but the expected time to build the corpus with our implementation was unacceptable. Thus, we simply added those comments that produced the highest increase in the vocabulary size to the collection by iterating through all the comments and checking the amount of increase in vocabulary size comment by comment. At each iteration, that comment with the highest contribution of new vocabulary to the final collection was selected. This process was repeated until 60,000 comments were reached.

3.3 Labeling Process

In order to establish the annotation schema, we followed those defined in (Wiegand and Siegel, 2018; Zampieri et al., 2019a), while introducing some additional details that we consider important. Namely, we created a new category to include those posts with inappropriate language but no offense intended. For instance, the comment “eres la puta ama” (you’re the fucking boss) contains inappropriate but non-offensive language and has a positive polarity. Then, we reformulated the definition of offensiveness to not include such posts.

The previous analysis led us to propose a definition of an offensive comment: one where language is used to commit an explicit or implicitly directed offense that may include insults, threats, profanity or swearing. Based on this definition, we established the following categories:

- **Offensive, the target is a person (OFP).** Offensive text targeting a specific individual.
- **Offensive, the target is a group of people or collective (OFG).** Offensive text targeting a group of people belonging to the same ethnic group, gender or sexual orientation, political ideology, religious belief, or other common characteristics.
- **Offensive, the target is different from a person or a group (OFO).** Offensive text where the target does not belong to any of the previous categories, e.g., an organization, an event, a place, an issue.
- **Non-offensive, but with expletive language (NOE).** A text that contains rude words, blasphemes, or swearwords but without the aim of offending, and usually with a positive connotation.
- **Non-offensive (NO).** Text that is neither offensive nor contains expletive language.

The annotation of the collected data was performed via Amazon Mechanical Turk (MTurk)\(^\text{10}\), which is a popular crowdsourcing platform. It provides the option of specifying some requirements that human annotators must meet to work on the task, and the time allotted per assignment. In our case, we selected the location as Spain and the time to five minutes due to the presence of some long comments from YouTube. Apart from releasing the annotation scheme with four examples of instances

### Table 1: Presence of offensive terms from lexicons in the retrieve comments.

<table>
<thead>
<tr>
<th>Social network</th>
<th>Offensive terms</th>
<th>Non-offensive terms</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>YouTube</td>
<td>19,449</td>
<td>184,414</td>
<td>203,863</td>
</tr>
<tr>
<td>Instagram</td>
<td>3,142</td>
<td>58,209</td>
<td>61,351</td>
</tr>
<tr>
<td>Twitter</td>
<td>1,197</td>
<td>18,728</td>
<td>19,925</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>23,788</strong></td>
<td><strong>259,865</strong></td>
<td><strong>283,622</strong></td>
</tr>
</tbody>
</table>

\(^{10}\)https://www.mturk.com/
for each class, in the purpose of ensuring clear and concise documentation, we also provided a list of instructions about rules, tips, and FAQs to try to solve any potential problems that could arise during the labeling process. Finally, to ensure the quality of the annotations, we used tracking comments.

We first conducted a round of trial annotation for both types of labeling, 4,500 and 1,500 instances with three and ten annotators, respectively. The goal of the trial annotation was (i) to identify any confusion in understanding the annotation schema, (ii) to estimate the average time to label the dataset, and (iii) to learn about the platform. The launch of these datasets was on September 24th, 2020, and it took two weeks to complete the annotation process on both sets. After analyzing the annotations, we observed through the comments of the annotators that the NOE and OFO classes were the most difficult to identify in the comments by the annotators. For this reason, we improved the definition of each class, providing examples as clear as possible to the annotators. The average agreement (kappa coefficient) grew from 36.85% for trial annotations up to 39.37% for final released comments. Yet, this level of agreement is lower than expected, which reflects the difficulty to discriminate among proposed classes.

Once the trial round was completed, the next step was to release the final dataset. A total of 54,023 instances were released in two subsets: 40,513 labeled by three annotators, and 13,510 labeled by ten annotators. The annotation took place from 17 November 2020 to 2 January 2021. As result, the three annotators subset covered 44,951 comments and the ten annotators subset 14,989 comments.

### 3.4 Post-processing

In order to check the reliability of the annotators, we analyzed their annotations in the tracking comments, i.e., those comments given as examples in the annotation guide. We observed that one of the annotators had over 60% of error rate in the tracking comments of both types of labeling, so we decided to remove their annotations since they could negatively affect the quality of the dataset. Sadly, this annotator was one of the most prolific, so the removal of his/her annotations resulted in a reduction of the three annotators subset to a number of 44,951 comments. A sample of the collected data is given in Tables 3 and 4 (Appendix).

### 3.5 Corpus Analysis

Thus, the final dataset is released divided into two subsets: the three annotators subset (3-Ann), with 44,951 comments, and the ten annotators subset (10-Ann), with 14,989 comments. The former is intended for multi-class classification research and the latter for tackling multi-output regression problems. Only 38 comments belong to both subsets. Comments are compiled without processing, therefore, case, punctuation, and emojis are preserved. Every comment is associated with a social network platform (Instagram, Twitter, or YouTube) and directed to one of the 12 selected influencers as the target. In Table 2, the amount of comments associated with each platform and influencer is depicted. Comments on dalas’ posts are more frequent (over 26% in both subsets). YouTube is the platform where most of the comments were collected (about 75% for both subsets), followed by Instagram (over 18%). Comments from Twitter only represent just over 6% of the collection.

For both subsets, the label is the majority class according to human annotators. For the subset labeled by ten annotators, the majority vote was set to five annotators. An additional None label was used when no agreement was reached between annotators. Table 3 shows the number of comments for each label on both subsets. Noticeably, the 10-Ann subset has a much lower percentage of None labels than the 3-Ann subset. The more annotators that were involved, the easier it was to decide the final label for a comment.

Table 4 shows statistics on comments length (i.e., the number of characters in the text). As expected, YouTube is the platform with the highest average length (about 190 for both subsets), with high variance; Twitter comments average length is lower (149 characters), with very small variance, and Instagram is the platform where comments tend to be the shortest (with an averaged length of 114).

Figure 1 shows the distribution of comments among influencers and social media platforms in the 3-Ann subset. YouTube is the most frequent platform, followed by Instagram. The influencer dalas is the target of more than a quarter of the total amount of comments. A similar distribution of comments is found in the 10-Ann subset.

An interesting analysis is to measure label frequency according to each influencer. Figure 2 shows the proportion of influencer-level labels and reflects the differences among these users as tar-
Table 2: Comments per social media and influencer in the OffendES dataset.

<table>
<thead>
<tr>
<th>Influencer</th>
<th>Instagram</th>
<th>Twitter</th>
<th>YouTube</th>
<th>Total</th>
<th>Instagram</th>
<th>Twitter</th>
<th>YouTube</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>dalas</td>
<td>3,558</td>
<td>1,454</td>
<td>6,813</td>
<td>11,825</td>
<td>(26.3%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>soyunapringada</td>
<td>582</td>
<td>31</td>
<td>5,412</td>
<td>6,025</td>
<td>(13.4%)</td>
<td>172</td>
<td>7</td>
<td>1,745</td>
</tr>
<tr>
<td>windygirk</td>
<td>466</td>
<td>487</td>
<td>3,756</td>
<td>4,709</td>
<td>(10.5%)</td>
<td>183</td>
<td>186</td>
<td>1,618</td>
</tr>
<tr>
<td>javioliveira</td>
<td>276</td>
<td>130</td>
<td>3,890</td>
<td>4,296</td>
<td>(9.6%)</td>
<td>92</td>
<td>52</td>
<td>1,297</td>
</tr>
<tr>
<td>wismichu</td>
<td>859</td>
<td>327</td>
<td>2,929</td>
<td>4,115</td>
<td>(9.2%)</td>
<td>318</td>
<td>101</td>
<td>1,014</td>
</tr>
<tr>
<td>miare</td>
<td>508</td>
<td>167</td>
<td>2,749</td>
<td>3,424</td>
<td>(7.6%)</td>
<td>166</td>
<td>63</td>
<td>936</td>
</tr>
<tr>
<td>wildhater</td>
<td>648</td>
<td>0</td>
<td>2,485</td>
<td>3,133</td>
<td>(7.0%)</td>
<td>204</td>
<td>0</td>
<td>843</td>
</tr>
<tr>
<td>nauterplay</td>
<td>540</td>
<td>0</td>
<td>2,058</td>
<td>2,598</td>
<td>(5.8%)</td>
<td>180</td>
<td>0</td>
<td>685</td>
</tr>
<tr>
<td>lauraescane</td>
<td>286</td>
<td>152</td>
<td>1,991</td>
<td>2,429</td>
<td>(4.6%)</td>
<td>107</td>
<td>50</td>
<td>633</td>
</tr>
<tr>
<td>dulceida</td>
<td>226</td>
<td>0</td>
<td>1,400</td>
<td>1,626</td>
<td>(3.6%)</td>
<td>81</td>
<td>0</td>
<td>440</td>
</tr>
<tr>
<td>jpelirrojo</td>
<td>69</td>
<td>0</td>
<td>582</td>
<td>651</td>
<td>(1.4%)</td>
<td>23</td>
<td>0</td>
<td>187</td>
</tr>
<tr>
<td>nosoymia</td>
<td>107</td>
<td>13</td>
<td>0</td>
<td>120</td>
<td>(0.3%)</td>
<td>42</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

| Total       | 8,125      | 2,761    | 34,065   | 44,951    | (18.6%)    | 2,791    | 955      | 11,243    |

Table 3: Comments per label in the OffendES dataset.

<table>
<thead>
<tr>
<th>Label</th>
<th>3-Ann</th>
<th>10-Ann</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>26,425</td>
<td>9,715</td>
</tr>
<tr>
<td>OFP</td>
<td>4,102</td>
<td>2,362</td>
</tr>
<tr>
<td>NOE</td>
<td>2,470</td>
<td>1,414</td>
</tr>
<tr>
<td>None</td>
<td>11,529</td>
<td>1,283</td>
</tr>
<tr>
<td>OFG</td>
<td>425</td>
<td>215</td>
</tr>
</tbody>
</table>

Table 4: Statistics over comments length.

<table>
<thead>
<tr>
<th></th>
<th>Average</th>
<th>Std. dev.</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>YouTube</td>
<td>189</td>
<td>247</td>
<td>3</td>
<td>9,986</td>
</tr>
<tr>
<td>Twitter</td>
<td>149</td>
<td>75</td>
<td>4</td>
<td>413</td>
</tr>
<tr>
<td>Instagram</td>
<td>114</td>
<td>124</td>
<td>3</td>
<td>2,200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Average</th>
<th>Std. dev.</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>YouTube</td>
<td>191</td>
<td>277</td>
<td>4</td>
<td>9,812</td>
</tr>
<tr>
<td>Twitter</td>
<td>150</td>
<td>74</td>
<td>5</td>
<td>292</td>
</tr>
<tr>
<td>Instagram</td>
<td>113</td>
<td>115</td>
<td>3</td>
<td>1,631</td>
</tr>
</tbody>
</table>

get of offensive comments. In terms of gender, it is can be seen that female influencers are subject to a greater number of offensive comments than male accounts. In particular, soyunapringada, miare_love, and WindyGirk are the accounts ranked with the most offensive comments. Regarding male influencers, accounts like JaviOliveira and NauterPlay contain more offense comments than accounts like WildHater and JPelirrojo. The profile of the influencer may define more controversy compared to others, or raise more negative emotions to their followers. Therefore, it could be interesting to consider the target profile as a source of information in offensive detection systems.

Inter-annotator agreement using the three annotators subset was measured with Cohen’s kappa coefficient. The k value is 0.3579 (fair agreement), which is quite low and reflects how difficult it is for humans to agree between the proposed categories. By analyzing annotations on tracking comments, we found that it was a common mistake to label a comment NOE or OFG when it should have been labeled OFO. Figure 3 shows the percentage of consensus per label in the subset of 3-Ann taking as consensus the majority vote (2-annotators agreement and 3-annotators agreement). As can be noticed, the label OFO exhibits the lowest consensus rate, with all three annotators only agreeing on 33.72% of the time. We found that many OFO comments were wrongly annotated with the NOE label and, actually, this could be reasonable since these offenses are not directly targeted to persons or groups, and they often consist in expletive ex-
pressions. Thus, we decided to merge them. After merging the OFO label into the NOE label, the kappa value increases slightly up to 0.3837. Figure 4 shows the final percentage of consensus per label after the merge of NOE and OFO labels.

Another feature we analyzed is the lexical diversity of comments. To this end, we use the MTLD metric already introduced, which allows us to get an insight into lexical variation and avoiding biases due to different text lengths. Table 5 shows the average values for MTLD for comments over labels and platforms, respectively.

As can be noticed, offensive comments targeted to a person (OFP) have low lexical diversity, as well as for those with expletive language (NOE). When the comment is not offensive at all, the lexical diversity is clearly higher. Regarding social networks, we would expect the lowest value of diversity in Twitter, as it limits comment length. On the contrary, Twitter is the platform with the highest lexical diversity, followed by YouTube. Instagram is clearly much poorer in terms of the diversity of vocabulary used. These findings are worth exploring, as they could provide more understanding of how language is used across platforms and how it relates to harmful language use, or on the average profile of their communities. To understand MTLD values, we have to consider that a value of 50 is the average lexical diversity of texts for an average adult text (being 80 for academic writings).

4 Baseline System

In order to establish a baseline for the OffendES corpus, we conducted experiments based on three different approaches:

Simple majority class model. Our simplest classifier assigns the majority class of the training set, i.e., the NO class, to each instance in the test set. This results in accuracy values of 58.78% and 64.85% respectively for 3-Ann and 10-Ann subsets.

Lexicon-based model. We also developed a lexicon-based approach using the lexical resources described in Section 3.2. In this approach, we only consider a binary classification scenario: whether the comment is offensive or not. For the 3-Ann subset, we obtained 67.13% of accuracy, 21.27% precision, 83.78% recall, and 33.93% F1. For the 10-Ann subset, the values of accuracy, precision, recall and F1 were, respectively, 71.45%, 35.59%, and 81.60%, 49.56%.

Transformer-based model. Finally, we experimented with a Spanish pre-trained BERT model called BETO (Canete et al., 2020) which has shown

<table>
<thead>
<tr>
<th>Social network</th>
<th>MTLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instagram</td>
<td>42.14</td>
</tr>
<tr>
<td>Twitter</td>
<td>61.74</td>
</tr>
<tr>
<td>YouTube</td>
<td>60.59</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Label</th>
<th>MTLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>66.36</td>
</tr>
<tr>
<td>NOE</td>
<td>26.41</td>
</tr>
<tr>
<td>None</td>
<td>53.59</td>
</tr>
<tr>
<td>OFG</td>
<td>53.19</td>
</tr>
<tr>
<td>OFP</td>
<td>28.68</td>
</tr>
</tbody>
</table>

Table 5: Average values of measures of lexical textual comments diversity per social network and label.
promising results in offensive language detection tasks (Plaza-del-Arco et al., 2020). Details about different configurations of the BETO model and the training process are given in the Appendix. In order to evaluate the model, we sampled from the collection two different sets, for training and evaluation. Measures used to report performance are Precision (P), Recall (R), and F1-score (F1) at class level, and macro and weighted average of these metrics. For the multi-output regression task, since we are not dealing with a multi-class scenario, we used one of the most preferred metrics for regression tasks, the mean squared error (MSE), a risk metric corresponding to the expected value of the squared (quadratic) error or loss.

### 4.1 Multi-class classification

This experiment is performed on the 3-Ann subset. All entries labeled as *None* were discarded (as no final label was assigned to these comments). The set was split into training (95%) and evaluation (5%) partitions, resulting in 30,079 comments in the training set and 3,343 in the evaluation set. Transformers (Wolf et al., 2020) library by Huggingface\(^\text{11}\) was used to build the BERT network and the tokenizer from available BETO models (uncased variant).

A sequence classifier was implemented for this multi-class task, with a final linear layer with four outputs (the logits for each possible label). Training the model took 2 hours and 26 minutes.

After seven training epochs, the model was evaluated against the evaluation partition. The results obtained are depicted in Table 6.

<table>
<thead>
<tr>
<th>Class</th>
<th>P (%)</th>
<th>R (%)</th>
<th>F1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>95.24</td>
<td>87.88</td>
<td>91.42</td>
</tr>
<tr>
<td>NOE</td>
<td>57.86</td>
<td>79.31</td>
<td>66.91</td>
</tr>
<tr>
<td>OFP</td>
<td>57.48</td>
<td>68.87</td>
<td>62.66</td>
</tr>
<tr>
<td>OFG</td>
<td>30.00</td>
<td>52.17</td>
<td>38.10</td>
</tr>
<tr>
<td>macro</td>
<td>60.15</td>
<td>72.06</td>
<td>64.77</td>
</tr>
<tr>
<td>weighted</td>
<td>86.96</td>
<td>84.39</td>
<td>85.33</td>
</tr>
</tbody>
</table>

Table 6: Multiclass experiment results.

### 4.2 Binary classification with BETO

Same configuration as the previous model, but using non-weighted cross-entropy as loss function during training. Classes have been merged into two classes as follows: *Non-offensive*, which comprises labels NO and NOE, and *Offensive*, combining OFP and OFG labels. This results in 28,895 non-offensive comments and 4,527 offensive comments. Training the model took 2 hours and 16 minutes. The results obtained are depicted in Table 7.

<table>
<thead>
<tr>
<th>Class</th>
<th>P (%)</th>
<th>R (%)</th>
<th>F1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-offensive</td>
<td>92.79</td>
<td>95.14</td>
<td>93.95</td>
</tr>
<tr>
<td>Offensive</td>
<td>68.06</td>
<td>58.33</td>
<td>62.82</td>
</tr>
<tr>
<td>macro</td>
<td>80.42</td>
<td>76.74</td>
<td>78.39</td>
</tr>
<tr>
<td>weighted</td>
<td>89.06</td>
<td>89.59</td>
<td>89.26</td>
</tr>
</tbody>
</table>

Table 7: Binary classification experiment results.

### 4.3 Multi-output regression with BETO

For every sample, a vector of probabilities is computed by counting the number of annotators that selected each label and dividing by the number of annotators. This provides an estimate of the confidence of each label to be assigned to the comment. Training the model took 48 minutes.

The 10-Ann dataset was split into training and validation partitions. After training for seven epochs over a partition of 13,020 samples, the model was evaluated against a partition of 685 test samples, obtaining an MSE of 0.0241.

### 5 Discussion

One of the main characteristics of the corpus is its imbalance at all levels: comments are not uniformly distributed across labels, influencers, or social platforms. The corpus size allows for stratified random sampling over those dimensions, but we considered that releasing the full set of comments is the best choice to allow researchers to decide on how to prepare their experiments. That is also the reason why comments with *None* class have been kept in the corpus, so different studies on the use of language within groups of young users of social networks can be conducted. Also, the *None* label is of interest by itself, as it reflects the absence of consensus in determining the nature of the comment.

Results show that deep learning models, like BERT, are good estimators of the presence of different kinds of offensive language, but that it is still a challenging task to decide whether a comment is directed to a person or not (so cyber-bullying risk could be measured). Despite the fusion of NOE

\(^{11}\)https://huggingface.co
and OFO categories, precision values for all labels different from NO are low.

6 Conclusion and Future Work

In this paper, we described OffendES: the first large-scale Spanish dataset of user comments on influencer posts from Instagram, YouTube and Twitter. It consists of 47,128 comments manually labeled for offensive content using a fine-grained annotation scheme. A subset of the corpus (10-Ann) assigns a confidence degree allowing both multi-class classification and multi-output regression studies. Additionally, a preliminary analysis of offensive behavior in social media and its relationship with the selected influencers is presented. Finally, baselines experiments have been performed, showing the validity of the corpus as well as the difficulty of the task.

A number of challenges remain open. On the one hand, we plan to explore systems trained on OffendES to monitor offensive messages in online channels participated by young people. On the other hand, the gender of the commenters and the subject of the comments have been left out for deeper analysis, so further research could be shed light on these matters. Finally, we believe that this dataset enables future work in the NLP community to tackle these interesting issues regarding Spanish language.

Acknowledgments

This work has been partially supported by a grant from European Regional Development Fund (FEDER), the LIVING-LANG project [RTI2018-094653-B-C21], and the Ministry of Science, Innovation and Universities (scholarship [FPI-PRE2019-089310]) from the Spanish Government.

References


E. Fersini, P. Rosso, and Maria Anzovino. 2018. Overview of the task on automatic misogyny identification at ibereval 2018. In IberEval@SEPLN.


A Appendix

A.1 Model settings

Hyper-parameters. In the experiments with Transformer the hyper-parameters used for fine-tuning BETO are specified in Table 1. In the multioutput regression task the hyper-parameters are the same, except for the loss function, which is replaced by mean squared error loss, as it is a regression problem.

All experiments (training and evaluation) were performed on a node equipped with two Intel Xeon Silver 4208 CPU at 2.10GHz, 192GB RAM, as main processors, and six GPUs NVIDIA GeForce RTX 2080Ti (with 11GB each).

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td>32</td>
</tr>
<tr>
<td>Epochs</td>
<td>7</td>
</tr>
<tr>
<td>Learning rate (LR)</td>
<td>2e-5</td>
</tr>
<tr>
<td>LR linear decrease</td>
<td>Yes</td>
</tr>
<tr>
<td>Loss</td>
<td>Weighted cross-entropy</td>
</tr>
<tr>
<td>Optimizer</td>
<td>AdamW</td>
</tr>
<tr>
<td>Weight-decay</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1: BETO fine-tuning hyper-parameters.

A.2 OffendES dataset

Table 2 shows the accounts used by the selected influencers in the three selected media: Instagram, Twitter, and Youtube.

Table 3 shows examples of labeled comments in the OffendES dataset by social network.
Table 2: Different account identifiers for selected influencers.

<table>
<thead>
<tr>
<th>Comment</th>
<th>Social Network</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 UNA MIERDA IGUAL QUE TU CANAL.</td>
<td>Instagram</td>
<td>OFO</td>
</tr>
<tr>
<td>2 El que llora siempre en sus videos por haber sido acosado para dar pena ahora acosa a gente... patético.</td>
<td>Twitter</td>
<td>OFP</td>
</tr>
<tr>
<td>3 El feminismo es cáncer y las feministas son mierda.</td>
<td>Youtube</td>
<td>OFG</td>
</tr>
<tr>
<td>4 Yo estoy de puta madre en casa... yo nací en cuarentena.</td>
<td>Youtube</td>
<td>NOE</td>
</tr>
<tr>
<td>5 Si pudiera viajar. Bueno iría a Italia. Que tengas un buen día saludos desde Buenos Aires, Argentina.</td>
<td>Instagram</td>
<td>NO</td>
</tr>
</tbody>
</table>

Table 3: Examples of comments labeled in OffendES (3-annotators subset), along with English translations.

<table>
<thead>
<tr>
<th>Comment</th>
<th>Social Network</th>
<th>OFP</th>
<th>OFG</th>
<th>OFO</th>
<th>NOE</th>
<th>NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Vieja ridícula.</td>
<td>Instagram</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2 Vaya tontería. Es campaña electoral, evidentemente unos le tiran mierda a los otros.</td>
<td>Twitter</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.7</td>
<td>0.3</td>
</tr>
<tr>
<td>3 Eres un cómico increíble siempre consigues sacarme una sonrisa y se me olvidan las penas.</td>
<td>Instagram</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4 Mocosos &quot;retrasados&quot;, ¿a alguien le ha sorprendido?, creo que no...</td>
<td>Youtube</td>
<td>0.1</td>
<td>0.7</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>5 Vaya mierda de video. Deja de hablar sin saber, gracias.</td>
<td>Youtube</td>
<td>0.3</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 4: Examples of comments labeled in OffendES (10-annotators subset), along with English translations.
On Machine Translation of User Reviews

Maja Popović, Andy Way
ADAPT Centre
School of Computing
DCU, Ireland

Alberto Poncelas
Rakuten Asia
Singapore
apontelas2@hotmail.com

Marija Brkić Bakarić
Department of Informatics
University of Rijeka, Croatia
mbrkic@uniri.hr

Abstract

This work investigates neural machine translation (NMT) systems for translating English user reviews into Croatian and Serbian, two similar morphologically complex languages. Two types of reviews are used for testing the systems: IMDb movie reviews and Amazon product reviews.

Two types of training data are explored: large out-of-domain bilingual parallel corpora, as well as small synthetic in-domain parallel corpus obtained by machine translation of monolingual English Amazon reviews into the target languages. Both automatic scores and human evaluation show that using the synthetic in-domain corpus together with a selected subset of out-of-domain data is the best option.

Separated results on IMDb and Amazon reviews indicate that MT systems perform differently on different review types so that user reviews generally should not be considered as a homogeneous genre. Nevertheless, more detailed research on larger amount of different reviews covering different domains/topics is needed to fully understand these differences.

1 Introduction

Machine translation (MT) has evolved very rapidly since the emergence of neural approaches in 2015, and it is being used for different genres and domains. Every year, evaluation campaigns which include both human and automatic evaluation are carried out with the goal of advancing the state of the art. The most well-known is the WMT shared task\(^1\) which focuses on news articles and (since 2016) on biomedical texts, and both can be considered as instances of “formal written text”. The IWSLT evaluation campaign\(^2\), on the other hand, focuses on the translation of TED talks, and some European projects (TraMOOC, transLectures) investigated the translation of online lectures. In both cases, the text can be considered to be “formal speech”, with the challenges of dealing with characteristics of spoken language and speech recognition output.

Recently, interest in the translation of user-generated content in the form of “informal written text” has been increasing. For example, JSALT 2019 workshop\(^3\) focused on translation of very noisy text content originating from sources like WhatsApp, Twitter and Reddit.

In this work, we focus on a different type of written user-generated content, namely user reviews. While the style is not as colloquial and noisy as that of Twitter or of other similar sources, it certainly is much less formal than news texts or other sources that have been investigated traditionally in the MT community. There are also important applications for focusing on this kind of data, both from commercial and from user perspective. More and more companies are expanding into multinational markets, and user reviews of products have become an important asset for online transactions and a feature that many customers expect to find. And in the era of always-available internet connectivity, many individuals rely on experiences of other people not only for guiding purchasing decisions, but also for entertainment options like choosing movies, books, restaurants, etc. In this work, we focus on both kinds of user reviews, namely product reviews from Amazon and movie reviews from IMDb.

Translating user reviews can increase and improve its reach and utility. The main issue for human translation is the fact that there is way too

---

\(^1\)http://www.statmt.org/wmt20/
\(^2\)http://workshop2019.iwslt.org/index.php
\(^3\)https://www.clsp.jhu.edu/workshops/19-workshop/improving-translation-of-informal-language/
much content to be translated. Therefore, MT is very helpful for this kind of content. However, the genre introduces several important challenges, such as informal language, spelling errors, a large number of domains/topics, and lack of in-domain parallel (bilingual) data.

In this work, we compare two approaches for building MT systems for translating user reviews: training on large parallel out-of-domain data and training on small synthetic in-domain data. We also compare MT performance on two types of user reviews: IMDb movies and Amazon products.

We investigate Croatian and Serbian as target languages, as a case involving mid-size less-resourced morphologically rich European languages. For these languages, a reasonable amount of out-of-domain parallel data is publicly available to train an NMT system, however still much lower than for “major” European languages (such as German, French, Spanish).

All our experiments were carried out on publicly available data sets. We used OPUS parallel data for out-of-domain training and a selected set of Amazon reviews for in-domain training. For development, we used the publicly available texts consisting of a selected set of English IMDb reviews and their Croatian and Serbian human translations. For testing, we used another selected set of IMDb reviews as well as a selected set of Amazon reviews. Neither of the test reviews has been investigated yet, and they will also be made publicly available.

1.1 Related work

A considerable amount of work in the Computational Linguistics/Natural Language Processing community has been done on processing user-generated content, mostly on sentiment analysis, but also on different aspects of machine translation (MT). Some papers investigate translating social media texts in order to map widely available English sentiment labels to a less supported target language (Balahur and Turchi, 2012, 2014).

Several researchers attempted to build parallel corpora for user-generated content in different language pairs in order to facilitate MT (Jehl et al., 2012; Ling et al., 2013; San Vicente et al., 2016), while (Banerjee et al., 2012) explored methods for domain adaptation. A recent JSALT Workshop dealt with improving MT for messages (Messenger, WhatsApp), social media (Facebook, Instagram, Twitter), and discussion forums (Reddit). Evaluating MT outputs of user-generated content was the topic of several publications, too. Two important measures of overall quality, comprehensibility and fidelity, were investigated in (Roturier and Bensadoun, 2011) in order to compare different English-to-German and English-to-French MT systems for technical support forums, and automatic estimation of these two measures for English-to-French MT was investigated in (Rubino et al., 2013). Maintaining sentiment polarity in German-to-English MT of Twitter posts was explored in (Lohar et al., 2017, 2018). However, none of these publications explored translation of user reviews.

The first publication about MT for user reviews (Lohar et al., 2019) explored translating English IMDb reviews into Croatian and Serbian and reported results of both automatic and human evaluation. However, all the systems were trained on very small amounts of parallel data so that the reported performance was rather low. More experiments on the same IMDb reviews were carried out (Popović et al., 2020), however, still only small amounts of training data were used. Also, no results of any kind of human evaluation were reported.

In this work, different sizes of the training corpora were explored, including a large corpus consisting of all publicly available parallel data for the two language pairs. Two types of reviews are explored, IMDb and Amazon, and both automatic scores as well as results of human evaluation are reported. In addition, differences between the two types of reviews are examined in order to see whether all user reviews can be considered as a homogeneous genre.

2 Building NMT systems

All our systems are based on the Transformer architecture (Vaswani et al., 2017) and built using the Sockeye implementation (Hieber et al., 2018). Previous work on the given two target languages (Popović et al., 2020) reported that multilingual sys-
tem which translates into both languages performs better than two separated bilingual systems. Therefore, all our systems are multilingual, built using the same technique as (Johnson et al., 2017; Aharoni et al., 2019), namely adding a target language label “SR” or “HR” to each source sentence. The amount of Croatian and Serbian data is balanced in all set-ups in order to achieve optimal performance for both target languages.

The systems operate on sub-word units generated by byte-pair encoding (BPE) (Sennrich et al., 2016b) with 32000 BPE merge operations both for the source and for the target language texts. We do not use shared vocabularies between the source and the target languages because they are distinct. On the other hand, we built a joint vocabulary for the two target languages because they are very similar.

All the systems have Transformer architecture with 6 layers for both the encoder and decoder, model size of 512, feed forward size of 2048, and 8 attention heads. For training, we use Adam optimiser (Kingma and Ba, 2015), initial learning rate of 0.0002, and batch size of 4096 (sub)words. Validation perplexity is calculated after every 4000 batches (at so-called “checkpoints”), and if this perplexity does not improve after 20 checkpoints, the training stops.

“Teacher/student” model As a first step, we built a system trained on all publicly available parallel data consisting of about 55 million sentences. These data, however, do not contain any user reviews. On the other hand, there is a vast amount of monolingual English user reviews, and in order to get use of it, we created a synthetic in-domain parallel corpus which is a widely used practice in NMT (Sennrich et al., 2016a; Zhang and Zong, 2016; Burlot and Yvon, 2018; Poncelas et al., 2018). We selected a set of about four million sentences from Amazon reviews originating from 14 different topics, and translated them by the system trained on out-of-domain data. In this way, we applied so-called “teacher/student” model, or “knowledge distillation” (Saleh et al., 2020; Chen et al., 2017; Kim and Rush, 2016). Knowledge distillation is the training of a smaller network (student) who learns from an already trained network (teacher). The idea is that the student will be performing much faster and hopefully approximately well as the teacher. The method is often used for reducing the amount of training data, to speed up the process, as well as for domain adaptation.

In our set-up, knowledge distillation is used for domain adaptation: the teacher model is the system trained on a large amount of out-of-domain parallel data. This system is used to create a small synthetic in-domain corpus, which is then used to train the student model.

“Advanced student” model The best option for using synthetic training corpora for NMT is not to use them alone, but to enrich “natural” parallel corpora. However, we do not have any natural in-domain parallel corpora. Yet, some parts of the large out-of-domain corpora might be more useful for translating reviews than others, especially subtitles which are usually informal spoken language. To explore this potential, we ranked out-of-domain sentences according to their similarity to user reviews, and extracted the most similar ones to combine them with the synthetic parallel corpus and train an “advanced student” model.

The details about all data sets and data selection are presented in the next section.

3 Data sets

3.1 User reviews

IMDb movie reviews (Maas et al., 2011) consist of about 10 sentences and 230 words on average. Each review is labelled with a score: negative reviews have a score < 4 out of 10, positive reviews have a score > 7 out of 10, and the reviews with more neutral ratings are not included.

In our experiments, IMDb reviews were used for development and testing, but not for training.

Amazon product reviews (McAuley et al., 2015) are generally shorter, consisting of 5 sentences and 93 words on average. Each review is labelled with a rating from 1 (worst) to 5 (best). The reviews are divided into 24 categories/topics/domains, and we used the reviews from the following 14 topics: “Beauty”, “Books”, “CDs and Vinyl”, “Cell Phones and Accessories”, “Grocery and Gourmet Food”, “Health and Personal Care”, “Home and Kitchen”, “Movies and TV”, “Musical Instruments”, “Patio, Lawn and Garden”, “Pet Supplies”, “Sports and Outdoors”, “Toys and Games”, and “Video Games”.

For our systems, Amazon reviews were used both for training as well as for testing, however
not for development. In order to obtain a balanced multi-target training corpus, half of the selected reviews from each of the topics were translated into Serbian and another half into Croatian.

3.2 Out-of-domain data

We used the publicly available OPUS\footnote{http://opus.nlpl.eu/} parallel data (Tiedemann, 2012) as out-of-domain data. The vast majority of these resources for the desired language pairs consists of OpenSubtitles, and there are also SETIMES News, Bible, Tilde, EU-bookshop, QED, and Tatoeba corpora. In addition, we used GlobalVoices for Serbian, and hrenWac, TED and Wikimedia for Croatian. In total, the corpus is well balanced over the two target languages.

3.3 Selected out-of-domain data

As mentioned in Section 2, we extracted a set of sentences from the out-of-domain subtitles according to their similarity to Amazon reviews. The subtitles were ranked using the Feature Decay Algorithm (FDA) (Biçici and Yuret, 2011, 2015; Poncelas et al., 2018; Poncelas, 2019). FDA selects sentences from a set $S$ based on the number of n-grams which overlap with an in-domain text $Seed$ and adds these sentences to a selected set $Sel$. In addition, in order to promote diversity, the n-grams are penalised proportionally to the number of instances already present in $Sel$. During the execution of FDA, candidate sentences from the set $S$ are selected one by one according to the following score:

$$\text{score}(s, Seed, Sel) = \frac{\sum_{ngr \in s \cap Sel} 0.5 C_{Sel}(ngr)}{\text{length}(s)}$$

The sentence $s$ with the highest score is removed from $S$ and added to $Sel$. The count of occurrences of n-gram $ngr$ in the selected set $Sel$, $C_{Sel}(ngr)$, is updated so that in the following iterations this n-gram contributes less to the scoring of one sentence. The process is executed iteratively, adding a single sentence from the set $S$ to the selected set $Sel$ at each step, and stopping after enough sentences have been extracted.

For our experiment, the out-of-domain subtitles represent the set $S$, and the Amazon reviews are $Seed$. From the 4 million English review sentences selected for training, we selected 140,000 sentences as seed (about 10,000 from each of the topics). We then used this seed to extract the similar sentence pairs from English-Croatian and English-Serbian subtitles. For each target language, we selected the top 9 million sentence pairs, thus 18M balanced sentence pairs in total.

Table 1 shows number of sentences, running words and distinct words (vocabulary) in training, development and test sets, as well as contributions of each of the review types.

4 Experimental set-up

In order to systematically explore influence of different sizes and natures of training data, we built the following MT systems:

- GENERAL (teacher model): system trained on all publicly available out-of-domain parallel data.
- REVIEWS (student model): system trained on in-domain synthetic corpus consisting of original English Amazon reviews and their translations generated by the GENERAL system.
- REVIEWS+SELECTED (advanced student): system trained on combination of synthetic in-domain data and selected natural out-of-domain data. We investigated different amounts of selected data:
  - REVIEWS+6M: adding 6 million selected out-of-domain sentences (3M for each target language)
  - REVIEWS+12M: adding 12 million selected out-of-domain sentences (6M for each target language)
  - REVIEWS+18M: adding all 18 million selected out-of-domain sentences (9M for each target language)

5 Results

5.1 Comparing MT systems

In order to get a quick feedback about each of our systems, we first evaluated them using the following three automatic overall evaluation scores: sacREBLEU (Post, 2018), chrF (Popović, 2015) and characTER (Wang et al., 2016).

The two best systems according to automatic scores, the “teacher” system GENERAL and the “advanced student” system REVIEWS+18M, were also evaluated by human annotators. The evaluators marked all words considered as adequacy errors, as described in (Popović, 2020), on a sub-set of about 200 sentences per system.
The results are presented in Table 2, and the tendencies are same for both target languages. As expected, the small synthetic in-domain corpus alone (REVIEWS) cannot achieve the same performance as the large out-of-domain corpus (GENERAL), however the difference in scores is not so large as could be expected considering the difference in the sizes (55M vs 4M) as well as the fact that the target part of the in-domain corpus is machine translated. Adding 6M of selected parallel sentences (REVIEWS+6M) slightly improves the performance, while additional 6M selected sentences (REVIEWS+12M) yield (and even slightly improve) the performance of the GENERAL “teacher” system. Adding 18M selected sentences (REVIEWS+18M) only slightly improves over the REVIEWS+12M system, and definitely outperforms the GENERAL “teacher” system. Since the improvements from 12M to 18M are rather small, we did not experiment with larger selected corpora.

We also present the scores for two on-line MT systems, AMAZON and GOOGLE, and it can be seen that our best two systems outperform both of them. Although their automatic scores are notably lower than the two best systems, they were also evaluated by human annotators in order to gather more annotations for comparing two different types of reviews which will be described in the next section.

Before moving to that, we will present a set of translation examples for the two best systems in Table 3. The first four sentences represent examples where the review-oriented “advanced student” system REVIEWS+18M performs better. In the sentence (1), the GENERAL system completely mistranslated the noun phrase “reddish brown hair”, and in the sentences (2) and (3) it choose incorrect variant of ambiguous source words “characters” and “care”. In the sentence (4), the word order is not optimal.

In sentences (5) and (6), REVIEWS+18M performed better on the first part of the sentence while GENERAL performed better on the second part. GENERAL failed to properly rephrase the first part of the sentence (5) and generated overly literal translation. In sentence (6), it choose incorrect variant of the ambiguous source word “great”. On the other hand, REVIEWS+18M failed to properly

Table 1: Data statistics: number of sentences, running words and distinct words (vocabulary) in training (a), development and test sets (b), and contribution (% of segments) of IMDb and Amazon reviews in training, development and test sets (c).
Table 2: Comparison of English→Croatian (a) and English→Serbian (b) systems trained on different texts by automatic evaluation scores: BLEU, chrF and characTER as well as by percentage of words marked as adequacy errors by human evaluators (“human”).

### 5.2 Comparing Amazon and IMDb reviews

In order to compare the MT performance of two types of reviews, separated scores for joint target languages are presented in Table 4. The reviews+18M system shows the best results for both types of reviews, which means that the “knowledge distillation” in form of forward translation of Amazon reviews by the general system was helpful for both review types.

Furthermore, for all systems, automatic scores are notably better for Amazon product reviews than for IMDb movie reviews, indicating that IMDb is more difficult for machine translation. However, the tendencies of human scores are different, except for Google. For other systems (our two and Amazon), the evaluators found less errors in IMDb than in Amazon reviews. Also, it has to be taken into account that IMDb reviewers were not used for training, only Amazon reviews, which can influence the results. More experiments with equal distributions in training and test sets should be carried out in future work.

After looking into errors marked by human evaluators in order to identify the most prominent error types (Popović, 2021), we found out that there are some differences in frequencies of certain error types, presented in Table 5. The largest difference can be seen for named entities, which are generally more frequent in IMDb than in Amazon reviews. Also, it has to be taken into account that IMDb reviewers were not used for training, only Amazon reviews, which can influence the results. More experiments with equal distributions in training and test sets should be carried out in future work.

...
| (1) source | Do not buy this unless you purposely want reddish brown hair. |
| reference | Ne kupujte ovo osim ako ciljano ne želite crvenkasto smeđu kosu. |
| GENERAL^- | Ne kupujte ovo, osim ako ne želite rashladenu kosu Reddish. |
| REVIEWS+18M^+ | Boring Characters |

| (2) source | Dosadni likovi |
| reference | Dosadni karakteri |
| GENERAL^- | Dosadni likovi |
| REVIEWS+18M^+ | Predivna briga za kožu. |

| (3) source | Wondervul Skin Care |
| reference | Predivna briga za kožu. |
| GENERAL^- | Predivna nega kože. |
| REVIEWS+18M^+ | Ovo je zapravo bio poprilično dosadan film. |

| (4) source | This was a pretty dull movie, actually. |
| reference | Ovo je zapravo bio poprilično dosadan film. |
| GENERAL^- | Ovo je zapravo bio poprilično dosadan film. |

| (5) source | I had high hopes for this product after reading all the wonderful reviews. |
| reference | Veliku nadu sam polagao u ovaj proizvod nakon čitanja svih tih divnih recenzija. |
| GENERAL^- | Najneverovatnija priča o ljudskoj dovitljivosti i kreativnosti! |
| REVIEWS+18M^+ | Najneverovatnija priča o ljudskoj genijalnosti i kreativnosti! |

| (6) source | A Great Story. The Most Amazing Tale of Human Ingenuity and Creativity! |
| reference | Sjajna priča. Najneverovatnija pripovetka o ljudskoj dovitljivosti i kreativnosti! |
| GENERAL^- | Sjajna priča. Najneverovatnija priča o ljudskoj genijalnosti i kreativnosti! |
| REVIEWS+18M^+ | Sjajna priča. Najneverovatnija priča X ljudske genijalnosti i kreativnosti! |

| (7) source | I don't like this kind of films, i feel like somebody is trying to pull my leg. |
| reference | Ne volim ovakve filmove, osjećam se kao da me netko pokušava prevariti. |
| GENERAL^- | Ne volim ovakve filmove, osjećam se kao da me netko pokušava prevariti. |
| REVIEWS+18M^- | Ne sviđa mi se ova vrsta filmova, osjećam se kao da me netko pokušava povući za nogu. |

| (8) source | My sense is that it depends to a large degree on the dog. |
| reference | Imam utisak da to mnogo zavisi od psa. |
| GENERAL^- | Moj osećaj je da to dosta zavisi od psa. |
| REVIEWS+18M^- | Moj osećaj je da to zavisi od velikog stepena na psa. |

| (9) source | I only recently discovered vanilla bean paste. |
| reference | Tek sam skoro otkrio pastu od zrna vanile. |
| GENERAL^- | Nedavno sam otkrio pastu od X vanile. |
| REVIEWS+18M^- | Nedavno sam otkrio pastu od vanile i pasulja. |

| (10) source | Horrifying Animal Cruelty |
| reference | Užasavajuća okrutnost prema životinjama |
| GENERAL^- | Zatrašujuća životinska okrutnost |
| REVIEWS+18M^- | Užasna Životinska Cruelty |

| (11) source | Poor Quality Cell Phone Charger |
| reference | Punjač mobitela loše kvalitete |
| GENERAL^- | Punjač s lošim kvalitetnim mobilnim telefonima |
| REVIEWS+18M^- | Siromašni punjač za mobitel |

Table 3: Translation examples for the two best systems, GENERAL and REVIEWS+18M. Errors together with the corresponding English parts are marked in bold. For the first four sentences, REVIEWS+18M is better; for (5) and (6), the two systems exhibit errors in different parts of the sentence; for (7), (8) and (9), GENERAL is better; for (10) and (11), both systems fail at the same part of the sentence.
<table>
<thead>
<tr>
<th>en→hr+sr</th>
<th>system</th>
<th>BLEU ↑</th>
<th>chrF ↑</th>
<th>cTER ↓</th>
<th>% of errors (human) ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Amazon</strong></td>
<td><strong>GENERAL</strong></td>
<td>57.8</td>
<td>68.7</td>
<td>26.5</td>
<td>15.1</td>
</tr>
<tr>
<td>products</td>
<td>REVIEWS+18M</td>
<td>58.2</td>
<td>69.2</td>
<td>26.1</td>
<td>14.0</td>
</tr>
<tr>
<td></td>
<td>AMAZON</td>
<td>56.6</td>
<td>67.9</td>
<td>26.8</td>
<td>21.4</td>
</tr>
<tr>
<td></td>
<td>GOOGLE</td>
<td>56.5</td>
<td>67.6</td>
<td>27.7</td>
<td>19.4</td>
</tr>
<tr>
<td><strong>IMDb</strong></td>
<td><strong>GENERAL</strong></td>
<td>49.1</td>
<td>63.3</td>
<td>31.6</td>
<td>12.9</td>
</tr>
<tr>
<td>movies</td>
<td>REVIEWS+18M</td>
<td>48.9</td>
<td>63.6</td>
<td>31.0</td>
<td>11.8</td>
</tr>
<tr>
<td></td>
<td>AMAZON</td>
<td>46.7</td>
<td>61.5</td>
<td>33.6</td>
<td>20.5</td>
</tr>
<tr>
<td></td>
<td>GOOGLE</td>
<td>44.2</td>
<td>58.2</td>
<td>38.0</td>
<td>23.6</td>
</tr>
</tbody>
</table>

Table 4: Comparison of automatic scores and human evaluation for two different types of reviews: Amazon products and IMDb movies. The scores are calculated on the joint test set for both target languages. All automatic scores are better for Amazon product reviews than for IMDb movie reviews, while the situation is different for human evaluation.

<table>
<thead>
<tr>
<th>error type (%)</th>
<th>IMDb</th>
<th>Amazon</th>
</tr>
</thead>
<tbody>
<tr>
<td>named entity</td>
<td>6.7</td>
<td>2.8</td>
</tr>
<tr>
<td>ambiguous word</td>
<td>10.9</td>
<td>12.9</td>
</tr>
<tr>
<td>gender</td>
<td>1.8</td>
<td>3.4</td>
</tr>
<tr>
<td>untranslated</td>
<td>0.9</td>
<td>2.5</td>
</tr>
<tr>
<td>non-existing word</td>
<td>0.7</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Table 5: Different error types in IMDb and Amazon user reviews; the largest difference can be noted for named entity errors, which are especially frequent in IMDb.

Among errors, untranslated words (English words copied into translation) as well as non-existing words (which do not exist either in the source or in the target language).

All these results indicate that there are differences between different types of reviews so that user reviews generally do not represent a homogeneous genre. However, the analysis is carried out on relatively small amount of data, especially human evaluation, so that it is not yet possible to draw any conclusions about the nature of these differences. Further analysis on more data as well as detailed analysis of different review topics including more review types (such as hotel reviews from TripAdvisor) should be carried out in future work.

6 Summary and outlook

This work investigates machine translation of two types of user reviews, IMDb movie reviews and Amazon product reviews, from English into Serbian and Croatian.

Since one of the main challenges for MT of user reviews is lack of parallel in-domain training data, we explored a possibility to make use of large out-of-domain bilingual parallel corpora as well as monolingual in-domain English corpora. We trained a general “teacher” system on all out-of-domain data and then used this system to create a small synthetic in-domain parallel corpus by translating English Amazon reviews into the target languages. Both automatic scores and human evaluation show that using this synthetic in-domain corpus together with a selected sub-set of out-of-domain data is the best option.

The results on separated IMDb and Amazon reviews indicate that MT systems perform differently on different review types so that user reviews generally should not be considered as a homogeneous genre. However, evaluating and training on larger amount of different reviews covering different domains/topics is needed to identify the nature of differences between different types of reviews, and also influence of different topics. Another direction of future work should include using more in-domain data, as well as other techniques for domain adaptation.

Acknowledgements

The ADAPT SFI Centre for Digital Media Technology is funded by Science Foundation Ireland through the SFI Research Centres Programme and is co-funded under the European Regional Development Fund (ERDF) through Grant 13/RC/2106. This research was partly funded by financial support of the European Association for Machine Translation (EAMT) under its programme “2019 Sponsorship of Activities”.

1120
References


Multilingual Coreference Resolution with Harmonized Annotations

Ondřej Pražák and Miloslav Konopík and Jakub Sido
{ondfa,sidoj}@ntis.zcu.cz
konopik@kiv.zcu.cz

Department of Computer Science and Engineering,
NTIS – New Technologies for the Information Society,
Faculty of Applied Sciences, University of West Bohemia, Technická 8, 306 14 Plzeň
Czech Republic

Abstract

In this paper, we present coreference resolution experiments with a newly created multilingual corpus CorefUD (Nedoluzhko et al., 2021). We focus on the following languages: Czech, Russian, Polish, German, Spanish, and Catalan. In addition to monolingual experiments, we combine the training data in multilingual experiments and train two joined models – for Slavic languages and for all the languages together. We rely on an end-to-end deep learning model that we slightly adapted for the CorefUD corpus. Our results show that we can profit from harmonized annotations, and using joined models helps significantly for the languages with smaller training data.

1 Introduction

Coreference resolution is the task of finding language expressions that refer to the same real-world entity (antecedent) of a given text. Sometimes the corefering expressions can come from a single sentence. However, the expressions can be one or more sentences apart as well. It is necessary to see the whole document in some hard cases to judge whether two expressions are corefering adequately. This task can be divided into two subtasks. Identifying entity mentions, and grouping the mentions together according to the real-world entity they refer to. The task of coreference resolution is closely related to anaphora resolution – see (Sukthanker et al., 2020) to compare these two tasks.

One of the challenging difficulties of coreference resolution lay in linguistically complicated annotations. Some examples of linguistic complications are split antecedents (a mention refer to more than one real-word entities), near identity relations, anaphoric and cataphoric relations, etc (Nedoluzhko et al., 2021).

In this paper, we rely on a CorefUD corpus (Nedoluzhko et al., 2021) of harmonized annotations. This corpus enables us to battle linguistic complications since it simply presents corefering mentions in clusters. Since the corpus is compiled from 11 different corpora in 8 different languages, we can conduct multilingual experiments in this work. Our research goal is to evaluate whether the harmonized annotations open the possibility to obtain some performance gain by joint learning on multiple languages. We aim to compare the harmonized annotations with the original corpora as well.

2 Related Work

In agreement with many other NLP tasks, deep learning models prevail in the coreference resolution task. Lee et al. (2017) were first to introduce the end-to-end approach that many following papers adopted (they obtained an average of 67.2 of F1 score). The task experienced a big leap in performance with the introduction of large pre-trained models. BERT based models deliver the best results; Kantor and Globerson (2019) F1 76.6, and Joshi et al. (2019) F1 76.9. Joshi et al. (2020) came up with a new pretraining task focused on better span representations. Their model called SpanBERT brings additional improvements in the coreference resolution task (F1 79.6). Xu and Choi (2020) question the importance of modeling higher-order inference (HOI). They show that with advanced encoders, HOI has only a minor effect on the performance of models.

Research is significantly less evolved for other languages than English. However, some notable experiments were published in recent years. Recasens et al. (2010) describe multilingual experiments (for English, Catalan and Spanish, Dutch, German and Italian) similarly to our paper. However, the annotations were not harmonized as in our case. Therefore, they provide no experiments with
joint training.

Other cross-lingual experiments include Portuguese by learning from Spanish (Cruz et al., 2018); Spanish and Chinese relying on an English corpus (Kundu et al., 2018); and Basque based on an English corpus as well (Urbizu et al., 2019). All these approaches employ neural networks, and they transfer the model via cross-lingual word embeddings.

Treex CR (Novák, 2017) is a coreference resolution module in the Treex NLP framework. It produces an advanced syntactic analysis with semantic features that the tool uses to find coreference relations – offers models for Czech and English. Other non-English experiments include Polish (Nitòn et al., 2018), Russian (Sboev et al., 2020), and German (Srivastava et al., 2018).

3 Dataset

For our experiments, we use the harmonized multilingual coreference dataset CorefUD (Nedoluzhko et al., 2021). The dataset was created by converting 17 existing datasets for 11 different languages into a common format on the top of universal syntactic annotations – Universal Dependencies. For coreference representation, a cluster-based approach was selected instead of the link-based approach. It is simpler and moreover the most frequently used dataset for English – OntoNotes adopt this approach too. In a cluster-based approach, every mention belongs to one cluster, represented by a unique ID. In a link-based approach, coreferences are expressed by the links between corefering mentions. In the link-based approach, coreference structures form a chain, but there are more complex coreference structures in some cases (Nedoluzhko et al., 2021). Datasets that use the link-based approach were converted to cluster-based at the cost of some information loss.

There are some notable differences between the datasets. One of the most prominent ones is the presence of singletons. Singletons are clusters that contain only one mention. Singletons are not present in any coreference relation. However, they are annotated as mentions in all datasets. Discontin-ued mentions represent another notable difference. A discontinuous mention consists of a sequence of words that is interrupted at least once with some words that do not belong to the mention. Such mentions can cause problems to models that assume mentions to be continuous (such as our model).

Table 1 shows the statistics of the datasets including the above-mentioned differences. More detailed analysis can be found in Nedoluzhko et al. (2021).

4 Model

We use the basic end-to-end model from Xu and Choi (2020) with no higher-order inference (HOI), so it is the same model as it was proposed by Lee et al. (2017).

In the model, we start by modeling the probability \( P(y_i|D) \) of a mention \( i \) corefering with the antecedent \( y_i \) in a document \( D \). Since the model adopts the end-to-end approach, the mentions are identified together with the coreference links. We consider every continuous sequence of words as a mention \( i \). Therefore, we work with \( N = \frac{T(T+1)}{2} \) possible mentions, where \( T \) is the number of words in a document \( D \).

We model the score of a mention \( i \) corefering with an antecedent \( y_i \) as a combination of two types scores \( s_m(i) \) and \( s_a(i, y_i) \). The \( s_m \) is a score of a sequence of words (spans) \( i \) being a mention. The \( s_a(i, y_i) \) score is the score of a span \( y_i \) being an antecedent of span \( i \). The scores are combined as a sum of \( s_m(i) \), \( s_m(y_i) \) and \( s_a(i, y_i) \) as follows:

\[
    s(i, y_i) = \begin{cases} 
    0 & y_i = \epsilon \\
    s_m(i) + s_m(y_i) + s_a(i, y_i) & y_i \neq \epsilon 
    \end{cases}
\]

(1)

where \( \epsilon \) is an empty antecedent. Both scores \( s_m(i) \) and \( s_a(i, y_i) \) are estimated with a feed-forward neural network over the BERT-based encoder. In our experiments we use two encoders – multilingual BERT (Devlin et al., 2018) and Slavic BERT (Arkhipov et al., 2019).

The probability of an antecedent \( y_i \) can be expressed as the \( \text{softmax} \) normalization over all possible antecedents \( y' \in Y(i) \) for a mention \( i \):

\[
    P(y_i|D) = \frac{\exp(s(i, y_i))}{\sum_{y' \in Y(i)} \exp(s(i, y'))} \tag{2}
\]

The formula for all antecedents uses a product of multinomials of all individual antecedents:

\[
    P(y_1, \ldots, y_N|D) = \prod_{i=1}^{N} P(y_i|D) \tag{3}
\]
In the training phase, we maximize the marginal log-likelihood of all correct antecedents:

\[
J(D) = \log \prod_{i=1}^{N} \sum_{\hat{y} \in Y(i) \cap \text{GOLD}(i)} P(\hat{y})
\]

where \text{GOLD}(i) is the set of spans in the training data that are antecedents.

5 Experiments

First, we perform monolingual experiments with the model described in Section 4 on several largest datasets from CorefUD. Namely Czech, Russian, Polish, Spanish, Catalan, and German-PotsdamCC. The employed datasets are summarized in Table 1 along with some basic statistics. The datasets are split to train, dev, and test, but the test datasets are not publicly available. Therefore, we use the original dev datasets as test datasets, and we create new dev datasets by taking 10% of the training parts. We tune the hyperparameters and perform early stopping on the development parts.

As the next step, we perform multilingual experiments, where we train one model for all the Slavic languages (Czech, Russian, and Polish) and another model for all the languages (Czech, Russian, Polish, German, Spanish, and Catalan). Multilingual results in comparison with the monolingual ones are shown in Table 2.

The results in Table 2 are influenced by the presence of singletons in the datasets. Particularly, Polish, German, Spanish, and Catalan contain a large portion of singletons, which negatively impacts the results. Since our employed model cannot model singletons, we have removed them from the test datasets. We show the results on filtered datasets in Table 4. Singletons are not important for coreference resolution since they form no coreference relation. However, they can be important in the training phase, if the model can use them for mention recognition.

We report the average F1 measure from the official scoring script for the coreference resolution task in CoNLL. The metric is computed as the average of MUC, B3 and CLEAF1. Definition of these metric can be found in Pradhan et al. (2014). The F1 scores are reported with 95% confidence intervals measured from 5 runs. We use the same set of hyperparameters for all the languages and for all models. We train the models for approximately 100k steps. We employ the Adam optimizer with the learning rate of 0.00001 for BERT layers and 0.0002 for other layers.

6 Discussion

From the result (See Table 3), we can see that joined multilingual models helps all the languages except for Czech – which does make sense because the Czech dataset is the largest one in the CorefUD corpus.

For the smallest datasets (Russian and German), multilingual models outperform the monolingual ones by a large margin (cca 2 – 6 % F1). We can see that at least in small training datasets, using joined models definitely helps, and we can profit from the harmonized coreference annotations. It is also worth noticing that the confidence intervals for these datasets are significantly wider than for other datasets.

After the singleton filtering the employed model achieves good results for all languages tested.

For German, there are 6.3% of discontinuous entity mentions. The model iterates over all possible continuous spans. Therefore, it is not able to identify discontinuous mentions at all. For German, the effect of singletons and discontinuous mentions

---

1 Table 1: Basic dataset statistics including train/dev/test split of CorefUD data sets. Column \text{discont.} shows the percentage of discontinuous mentions. Taken from Nedoluzhko et al. (2021).

2 https://github.com/conll/reference-coreference-scorers

---

1125
Table 2: Overall results of F1 averages obtained from the official scoring script.

<table>
<thead>
<tr>
<th>Model</th>
<th>czech</th>
<th>russian</th>
<th>polish</th>
<th>german</th>
<th>spanish</th>
<th>catalan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mono-mBERT</td>
<td>58.883±0.204</td>
<td>62.665±1.028</td>
<td>42.411±0.303</td>
<td>39.958±0.775</td>
<td>49.654±0.118</td>
<td>47.962±0.302</td>
</tr>
<tr>
<td>Mono-SlavicBert</td>
<td>60.283±0.013</td>
<td>62.097±1.153</td>
<td>43.234±0.114</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Slavic-mBERT</td>
<td>58.734±0.198</td>
<td>66.762±0.495</td>
<td>44.091±0.413</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Slavic-SlavicBERT</td>
<td>60.096±0.103</td>
<td>64.414±0.750</td>
<td>44.943±0.110</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Joined-mBERT</td>
<td>58.990±0.304</td>
<td>65.243±0.942</td>
<td>44.346±0.342</td>
<td>46.098±0.641</td>
<td>51.192±0.221</td>
<td>49.881±0.126</td>
</tr>
</tbody>
</table>

Table 3: F1 gains of multilingual models over the same monolingual ones. Bold numbers indicate that the difference is bigger than the width of confidence interval. Table depicts absolute differences.

<table>
<thead>
<tr>
<th>Model</th>
<th>czech</th>
<th>russian</th>
<th>polish</th>
<th>german</th>
<th>spanish</th>
<th>catalan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joined-mBERT</td>
<td>+0.107</td>
<td>+1.926</td>
<td>+1.935</td>
<td>+6.140</td>
<td>+1.538</td>
<td>+1.919</td>
</tr>
<tr>
<td>Slavic-mBERT</td>
<td>-0.149</td>
<td>+3.445</td>
<td>+1.680</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Slavic-SlavicBERT</td>
<td>-0.187</td>
<td>+2.317</td>
<td>+1.709</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4: Overall results of F1 averages obtained from the official scoring script after singleton removal.

<table>
<thead>
<tr>
<th>Model</th>
<th>czech</th>
<th>russian</th>
<th>polish</th>
<th>german</th>
<th>spanish</th>
<th>catalan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mono-mBERT</td>
<td>64.383±0.153</td>
<td>63.135±0.521</td>
<td>60.247±0.242</td>
<td>52.541±1.183</td>
<td>67.88±0.543</td>
<td>64.394±0.685</td>
</tr>
<tr>
<td>Mono-SlavicBert</td>
<td>65.835±0.141</td>
<td>63.453±0.615</td>
<td>61.726±0.395</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Slavic-mBERT</td>
<td>63.980±0.211</td>
<td>66.794±1.105</td>
<td>61.584±0.396</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Slavic-SlavicBERT</td>
<td>65.443±0.231</td>
<td>64.192±0.475</td>
<td>62.883±0.068</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Joined-mBERT</td>
<td>64.176±0.120</td>
<td>65.618±0.314</td>
<td>61.959±0.431</td>
<td>61.439±1.216</td>
<td>68.9825±0.209</td>
<td>66.456±0.092</td>
</tr>
</tbody>
</table>

7 Future Work

Currently, we experimented only on a subset of languages available in CorefUD. This was caused mainly by the resource-exhaustive training. We need 32GB graphic cards to capture long-enough contexts. We plan to experiment with the rest of the languages in the future.

Additionally, it would be interesting to explore the possibilities of zero-shot cross-lingual transfer in CorefUD, where we will not use the training data for the target language at all.

8 Conclusion

We performed pilot experiments to evaluate inter-language transferability of the models based on the CorefUD dataset. To do so, we used an end-to-end coreference resolution model based on BERT-like models. Our experiments show that learning from multiple languages in CorefUD annotation scheme helps significantly especially for languages with smaller training data (like Russian and German data in CorefUD).

Acknowledgments

This work has been partly supported from Grant No. SGS-2019-018 Processing of heterogeneous data and its specialized applications. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

References


Predicting Informativeness Of Semantic Triples

Judita Preiss
University of Salford
Salford M5 4WT, United Kingdom
j.preiss@salford.ac.uk

Abstract

Many automatic semantic relation extraction tools extract subject-predicate-object triples from unstructured text. However, a large quantity of these triples merely represent background knowledge. We explore using full texts of biomedical publications to create a training corpus of informative and important semantic triples based on the notion that the main contributions of an article are summarized in its abstract. This corpus is used to train a deep learning classifier to identify important triples, and we suggest that an importance ranking for semantic triples could also be generated.

1 Introduction

Subject-predicate-object triples are used in numerous natural language processing areas, including question answering (e.g. Hristovski et al. (2015)), ontology building (e.g. Du and Li (2020)) and literature based discovery (e.g. Hristovski et al. (2006)). While they can be thought of as representing the minimum unit of semantic expression, there is a large degree of variability in the amount of new (not commonly known) content they convey. On the one hand, they sometimes represent what can be termed background knowledge, for example “New Zealand - ISA - country” or “pharmaceutical services - TREATS - health personnel”, while on the other, they may describe very specific findings such as pimobendan TREATS hypertrophic cardiomyopathy or LCN2 protein, human - ASSOCIATED_WITH - chronic kidney disease. We use biomedical publications to test the hypothesis that training data consisting of such, important, triples can be created from abstracts, and train a deep learning algorithm to identify these high importance triples from a list of all triples appearing in a paper. The system could also be adjusted to output a weight instead of a binary decision, allowing for an importance ranking of semantic triples within an article.

The paper begins with an overview of related work in Section 2, the experimental set-up follows in Section 3, with the results and discussion in Section 4 and conclusions drawn in Section 5.

2 Background

A number of tools for automatically extracting semantic relations – (subject, relation, object) triples – from unstructured text exist (Yuan and Yu, 2018). However, as Papadopoulos et al. (2020) point out, the majority of works incorporating these do not perform much pre- or post-processing and therefore include many potentially uninformative triples, and works proposing to extend currently existing collections of semantic relations often speak of extending the set of relations, not refining the relations present (e.g. Koroleva et al. (2020)).

Evaluations of semantic relation extraction systems are often very comprehensive, e.g. Kilicoglu et al. (2020) present a detailed independent evaluation of SemRep – a biomedical domain tuned triple extraction tool – and discover common sources of error for this tool, but such evaluations do not quantify the quality of the triple that is retrieved by the system. It is unclear whether the incorrectly extracted triples are uninformative, or the opposite.

While not phrased as focusing on informative / important triples, existing works often restrict to particular types of relations: Yuan and Yu (2018) evaluate the extraction of health claims, defined as a relation between something that is being manipulated and something that is being measured (e.g. the relation between a substance and a disease). Yadav et al. (2020) restrict to drug-drug interaction, protein-protein interaction, and medical concept relation extraction, while Hope et al. (2021) focus on mechanisms, i.e. activities, functions and...
causal relations. Such restrictions are likely to increase the overall quality of the remaining triples: removing the ISA relation alone eliminates a large quantity of background knowledge. The closest to our work is due to Zhang et al. (2021) who filter out uninformative triples computationally, based on the difference between triples’ expected and observed frequencies.

3 Experiment Design

Below, we discuss the two steps needed to explore the hypothesis that a dataset based on abstracts can be used to detect important triples using machine learning: 1) creation of a training corpus, and 2) selection of a deep learning architecture.

3.1 Training Corpus Creation

The CORD-19 dataset (Wang et al., 2020) was chosen for this work due to: 1) scale, the 2021-05-03 version contains 198,347 full articles, 2) availability of extracted text, the dataset contains the text extracted from available full article PDFs, 3) domain, the restricted nature of the dataset allows the application of existing biomedical tools.

3.1.1 Semantic Relation Extraction

Subject-relation-object triples are extracted from all article texts present in the dataset using SemRep (Rindflesch and Fiszman, 2003). Designed for the biomedical domain, the tool extracts triples such as “*imatinib* TREATS Gastrointestinal Stromal Tumors” but with concepts mapped to Unified Medical Language System metathesaurus (UMLS) (Bodenreider, 2004) concept unique identifiers, CUIs (i.e. yielding C0935989 - TREATS - C0238198 for the example). This addresses the problem of multi-word identification (recognizing gastrointestinal stromal tumours rather than merely tumours) and word sense disambiguation (distinguishing between occurrences of concepts with multiple meanings, such as COLD, which could - among other options - represent the common cold or chronic obstructive airway disease).

3.1.2 Identifying Important Triples

To train a machine learning classifier, a training set of important triples is needed. Since an abstract usually summarizes the main findings of an article, we hypothesize that important triples can be considered to be those that appear in both the body and an abstract. It is important to note that the training set of important triples does not need to be complete, i.e. not every important triple from the body needs to be identified. The dataset should be as noise free as possible, and therefore background knowledge triples (which may appear in both the abstract and the body of an article) should not be included. To reduce noise, the following filtering is performed:

- Previously published triples. The construction of positive examples in the training set hinges on the identification of important triple(s). If these triples are defined as those which describe the novel contribution(s) of an article, an identical triple (i.e. contribution) should not have appeared in abstracts prior to the current paper. Therefore triples appearing in SemRep processed Medline (V40, released October 2019, i.e. before the CORD-19 dataset), a vast collection of biomedical abstracts (Lozano-Kühne, 2013), are removed from the dataset.
- Frequent concepts. Some frequent concepts often appear in non important triples, such as: 
  - therapeutic procedure TREATS disease
  - malaise PROCESS_OF patients
  - lung PART_OF *homo sapiens*

Since the training set does not need include an annotation for every triple encountered and there is high probability of mis-annotation with triples involving these concepts, triples involving the top 1% of concepts appearing in V40 of SemRep processed Medline are removed. The top 1% includes patients, therapeutic procedure, homo sapiens and other very general terms. Note that this does not mean that the system will be unable to classify triples including these concepts.

In some cases, an identical triple is used both in the abstract and the body of an article, however, when repeated, novel contributions of a paper are sometimes rephrased using (near) synonyms. Therefore a measure of triple similarity needs to be defined. Since the triples are of the format *subject CUI* → *predicate word* → *object CUI*, this measure can be defined on each component (subject, predicate, object) separately. Word (CUI) embeddings represent each word (CUI) as a vector which captures information about the contexts it appears in, therefore yielding similar – close – vectors for synonyms. A triple similarity measure can therefore be implemented based on cui2vec (Beam et al.,
2019) (for subject and object similarity) and GloVe embeddings (for predicate similarity).

Similarity between two triples, \( \text{cui}_1 - \text{rel}_1 - \text{cui}_2 \) and \( \text{cui}_21 - \text{rel}_2 - \text{cui}_22 \), is then given by the formula:

\[
\text{cs}(\text{c}2\text{v}(\text{cui}_1), \text{c}2\text{v}(\text{cui}_21)) + \text{cs}(g(\text{rel}_1), g(\text{rel}_2)) + \text{cs}(\text{c}2\text{v}(\text{cui}_12), \text{c}2\text{v}(\text{cui}_22))
\]

where \( \text{cs} \) represents the cosine similarity, \( \text{c}2\text{v}(x) \) the cui2vec vector of \( x \) and \( g(x) \) \( x \)'s GloVe vector.

As the maximum value for cosine similarity is 1, the triple similarity is a decimal between 0 and 3 inclusive, 0 corresponding to complete lack of similarity between triples and 3 an exact match. For each body-triple, a similarity can be computed between it and each abstract-triple in the same article, with the highest becoming the body-triple’s similarity value. A threshold can be set on the similarity value to decide which triples are deemed important.

3.2 Deep Learning Algorithm

The machine learning component consists of three parts: 1) feature extraction, 2) architecture selection, and 3) experiment settings.

3.2.1 Feature Extraction

The ability to extract important triples (described in Section 3.1.2) makes it possible to use supervised machine learning approaches to train a classifier. To this end a number of features are extracted for each body-triple.

Frequency based features: 1) the number of times the triple appeared in the body of the article, and 2) the total number of relations within the body of the publication.

UMLS based features: 1) the frequency count of the CUIs in the body triple as extracted from SemRep processed Medline – while the top 1% of CUIs have been discarded, it is believed that CUIs with lower frequencies are more likely to be part of novel contributions, 2) the UMLS source vocabulary of the CUIs – the metathesaurus consists of many different types of biomedical vocabularies and the information pertaining to which one(s) a CUI belongs to can serve to give an overall idea of its category, and 3) the depth of the body triple CUIs within UMLS. For some source vocabularies, a hierarchy is present, allowing the computation of the concept’s distance to the root – assuming a concept further away from the root is more likely to be more fine-grained, this feature also investigates whether important triples are more likely to contain more specific CUIs (the shortest path to the root is taken if a concept appears in multiple hierarchies).

Semantics based features: 1) the relation used, 2) the title of the section the body triple appeared in – since the majority of articles in this collection have relatively rigid structure, this was restricted to the commonly prescribed sections such as introduction, background, methods etc, and is based on the hypothesis that a novel contribution of a work is likely to appear in the discussion and / or conclusion sections, and 3) the rank of the sentence the triple appeared in as ranked by TextRank (Mihalcea and Tarau, 2004). TextRank is a graph based algorithm, often used in summarization, which can be used to order the sentences in an article according to importance, and therefore we hypothesize that a sentence with a low TextRank (high importance) is more likely to yield an important triple.

After performing one hot encoding of the relation feature, this gives 129 features for the 55,745 triples in the dataset.

3.2.2 Architecture Selection

While the similarity value of a body-triple calculated as described in Section 3.1.2 can be predicted directly, initial experiments with regression showed that this is hard to do exactly. The problem was therefore framed as binary classification. In this case, a threshold is set on the similarity value and triples with a value above the threshold are used as positive, important, instances.

Deep learning model is chosen due to its ability to cope with feature dependencies. The model, implemented using Keras, was designed with fully connected (dense) layers of halving sizes with the final layer of size 1. ReLU was used for all layers except the last, where the sigmoid activation function was employed. The loss function was binary entropy and accuracy was used as the metric when classes weren’t extremely imbalanced, \( F_1 \) was used otherwise. A number of parameters were tuned: 1) the depth of the model (with halving sizes, thus depth one model has a single dense layer of size \( \text{int}(129/2) \), depth two model has two dense layers of sizes \( \text{int}(129/2) \), \( \text{int}(129/4) \), and so on), 2) the number of epochs, 3) dropout, and 4) whether class weights were used.

3.2.3 Experiments

As suggested above – by exploring the use of class weights within the model – the dataset is highly imbalanced with, as expected, the majority of triples
not appearing in the abstract. The following methods for addressing this bias were explored:

- Using class weights within the deep learning algorithm: this allows more emphasis to be given to the minority class.

- Under-sampling: randomly sampling the majority class such that the number of examples used in training corresponds to a pre-decided ratio. The minority and majority class can be made equal (1:1) but other ratios were explored, making the majority class more frequent but not overpowering.

While all the minority, important, class triples are included in the training set, this does not have to be the case for the majority class. As mentioned above, the triples to include in the minority class are selected by a threshold. However, this can lead to a triple with, say, similarity of 2.5 being included in important triples, while a triple with similarity of 2.499 appearing in the non important triples class. Such small difference may be detrimental to the performance of the machine learning algorithm and a buffer band of similarities between the two classes was also explored. I.e. two thresholds, \( t_1 \) and \( t_2 \) are set such that \( t_1 - t_2 > 0 \) and all triples with similarity \( \geq t_1 \) are assigned to the important class while triples with similarity \( \leq t_2 \) are deemed not important.

### 4 Results And Discussion

A 5-fold cross validation was performed, and each explored model was trained on (a possibly balance adjusted version of) the training portion giving rise to an accuracy or F-measure on the test portion. This allows an average to be computed and the best model to be determined. The results are presented in Table 1: the similarity value refers to the threshold from Section 3.1.2 used to determine which triples are considered important, the buffer band – when on – removes the cases close to the similarity value threshold from training as described in Section 3.2.3, and the majority column represents the percentage of the training dataset attributed to the majority class. The final columns present the hyperparameters of the best model for the specific combination and the average accuracy / F-measure.

With under-sampling, the accuracies for similarities \( \geq 2 \) were all within 2% of the best performance, supporting the hypothesis regarding frequent use of synonyms. To avoid a uniform assignment of the majority class, the F-measure metric (which rewards both precision and recall) is used in models without under-sampling. An F-measure of 1 represents perfect precision and recall, and the highest F-measure achieved is 0.975.

SHapley Additive exPlanations (SHAP) (Lundberg et al., 2018) uses ideas from game theory to explain feature contributions to machine learning decisions. Figure 1 depicts the feature contributions on a randomly selected sample of 100 triples for the best model without under-sampling. Each dot represents a single triple, with the intensity (blue \( \rightarrow \) pink) indicating whether the feature value was low or high. The horizontal position indicates whether the contribution caused the prediction to go up – towards being classified as an important triple – or down. The top three rows show expected results: that high values in the number of relations in the document, very frequently occurring CUIs or relations arising from sentences low in importance ranked by TextRank (giving a high rank) impact the prediction very negatively. Unsurprising positive contributors are: 1) the frequency of the triple in the document: a new contribution may be reiterated in the document, 2) the triple appearing in the conclusion: this often contains a summary of contributions, 3) the triple including the TREATS relation: the filtering ensures this is a new triple and being treatment specific, is likely the focus of the work, 4) the triple appearing in the intro-

<table>
<thead>
<tr>
<th>Similarity Value</th>
<th>Buffer Band</th>
<th>Majority Proportion</th>
<th>Best Model</th>
<th>Accuracy / F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \geq 3 )</td>
<td>ON</td>
<td>50</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>( \geq 3 )</td>
<td>OFF</td>
<td>50</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>( \geq 3 )</td>
<td>ON</td>
<td>83.3</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>( \geq 3 )</td>
<td>OFF</td>
<td>83.3</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>( \geq 2 )</td>
<td>OFF</td>
<td>88.3</td>
<td>3</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 1: Performance of informative triple classifier
duction, where the novelties of the work are often highlighted. The contributions of a higher depth value is also, as expected, positive.

Contributions are also due to the CUIs’ UMLS source vocabulary (indicated by source_). In some cases, these categorize the CUI; for example, AOD (alcohol and other drug thesaurus) and PSY (psychological index terms) are not unexpected. Surprising may be the pair MSHNOR and MSHJPN, representing the Norwegian and Japanese translations of Medical Subject Headings, as they appear to have opposite effect. However, MSHJPN’s contribution is very limited, suggesting that its completeness may not match that of MSHNOR.

5 Conclusions And Future Work

We have demonstrated that a dataset of semantic triples created from full articles based on similarity between triples in the body of the text and triples in the abstract can be used to train a deep learning classifier to make predictions about a semantic triple’s importance. An analysis of feature contributions was also performed.

While a direct prediction of the similarity score appeared difficult with the quantity of data available, converting the similarity scores into categorical values may be trainable and would provide the basis of a ranking. Again with greater quantity of data, features based on medical subject headings of each CUI could be beneficial indicated by the success of the UMLS source vocabulary features.

The work undertaken was in the biomedical domain based on a tool tuned for biomedical domain grammatical relation extraction. Porting the approach to another domain, where subject-verb-object triples would need to be extracted using a generic grammatical relation extraction algorithm and some features would require re-engineering, would also form an interesting extension of the work.

Acknowledgements

This work was partly supported by the ERDF Greater Manchester AI Foundry grant, and I would like to thank Sunil Vadera in particular for his support. Thanks also go to the anonymous reviewers for ensuring clarity of this paper as well as pointing out some very nice avenues for further work.

References


Dimitar Hristovski, Dejan Dinevski, Andrej Kastrin, and Thomas Rindflesch. 2015. Biomedical question
answering using semantic relations. *BMC Bioinformatics*, 16(6).


Unknown Intent Detection Using Multi-Objective Optimization on Deep Learning Classifiers

Prerna Prem¹, Zishan Ahmad¹, Asif Ekbal¹, Shubhashis Sengupta², Sakshi C. Jain², Roshni Ramnani²
¹Department of Computer Science and Engineering, Indian Institute of Technology Patna
²Accenture Technology Lab, Bangalore
prernaprem21@gmail.com 1821cs18@iitp.ac.in
asif.ekbal@gmail.com shubhashis.sengupta@accenture.com
sakshi.c.jain@accenture.com roshni.r.ramnani@accenture.com

Abstract

Modelling and understanding dialogues in a conversation depends on identifying the user intent from the given text. Unknown or new intent detection is a critical task, as in a realistic scenario a user intent may frequently change over time and divert even to an intent previously not encountered. This task of separating the unknown intent samples from known intents one is challenging as the unknown user intent can range from intents similar to the predefined intents to something completely different. Prior research on intent discovery often consider it as a classification task where an unknown intent can belong to a predefined set of known intent classes. In this paper we tackle the problem of detecting a completely unknown intent without any prior hints about the kind of classes belonging to unknown intents. We propose an effective post-processing method using multi-objective optimization to tune an existing neural network based intent classifier and make it capable of detecting unknown intents. We perform experiments using existing state-of-the-art intent classifiers and use our method on top of them for unknown intent detection. Our experiments across different domains and real-world datasets show that our method yields significant improvements compared with the state-of-the-art methods for unknown intent detection.

1 Introduction

Detecting whether an intent is unknown or new in a dialogue system has become an important task for improving customer satisfaction. Since user intent may frequently change over time in many realistic scenarios, unknown (new) intent detection has become a crucial problem in conversational artificial intelligence (CAI). This can ultimately help enhance system interaction with the customer. This task is challenging since there is no prior knowledge of the type or the exact numbers of unknown intents that would be encountered in the future.

We model unknown intent detection as an (m+1)-class classification task as suggested by (Shu et al., 2017; Lin and Xu, 2019; Zhang et al., 2020) and consider unknown classes as the (m+1)th class. We aim to identify the known intent samples accurately, while at the same time we focus on determining the unknown intent samples. This has to be done without any prior knowledge about the kind of unknown intents. In order to solve this problem, researchers have proposed deep neural networks like OpenMax (Bindale and Boult, 2016), which fits Weibull distribution to the outputs of the penultimate layer. Another system MSP (Hendrycks and Gimpel, 2016) calculates the softmax probability of known samples and discards the unknown samples with lower confidence. In our approach, we attempt to solve the problem of unknown intent detection with added constraints such as not having any prior knowledge of a finite set of intents.

The main contributions of this paper are:

1. We propose an efficient method for unknown intent detection that post-processes using multi-objective optimization (non-deterministic genetic algorithm-NSGA2) by optimising two objectives i.e. recall and precision in order to obtain the optimal thresholds for each intent class.

2. Our proposed weight fine-tuning approach is model-agnostic, i.e. it can be applied on top of any deep neural network model.

The rest of the paper is organized as follows. In Section 2, we present the literature survey of previous work done on this topic. In Section 3 we elaborate describe the proposed methodology. In Section 4, we discuss the experimental setup and the datasets used in our experiments. In Section
5. we analyse the results of detecting the unknown intents. Finally, Section 6 concludes the paper with future work that can be explored in this field.

2 Related Works

Intent detection is a much explored area in dialogue systems with a broad spectrum of literature available (Min et al., 2020; Qin et al., 2020; Zhang et al., 2018; Niu et al., 2019; Qin et al., 2019). Most of these works are based on closed world classification that does not consider any open intent. (Srivastava et al., 2018) proposed a zero-shot learning (ZSL) for intent detection. However, ZSL is different from our task as it only contains finite known set of classes during testing. (Kim and Kim, 2018) tried to optimise the intent classifier together with an out-of-domain detector, which was trained using out-of-domain samples. The generative method proposed by (Yu et al., 2017) used adversarial learning to generate positive and negative examples from known classes but the method did not work well in the discrete data space like text. (Ryu et al., 2018) proposed generative adversarial network (GAN) to train on the ID samples and use the discriminator to detect the out-of-domain samples. (Nalisnick et al., 2018; Mundt et al., 2019) showed that deep generative models fail to capture the high-level semantics on real world data. (Jain et al., 2014) fit the probability distributions to statistical Extreme Value Theory (EVT) using a Weibull-calibrated multi-class support vector machine (SVM) to detect the unnormalized posterior probability of inclusion for open set problems. ODIN (Liang et al., 2017) enlarged the differences between known and unknown samples by using temperature scaling and input pre-processing but all the above method need negative samples for selecting the decision boundary or probability threshold. DOC (Shu et al., 2017), instead of using Softmax as the final output layer, built a multi-class classifier with a 1-vs-rest final layer which contains a sigmoid function for each seen class to reduce the open space risk.

Zero-shot intent classification aims to generalize knowledge and concepts learned from the seen intents to recognize unseen intents. Early methods (Ferreira et al., 2015a,b) explored the relationship between seen and unseen intents by introducing external resources such as manually defined attributes or label ontologies, but they are usually expensive to obtain. To deal with this, some methods (Chen et al., 2016; Kumar et al., 2017) map the utterances and intent labels to an embedding space and then model their relations in the same space. IntentCapsNet-ZS (Xia et al., 2018) extends capsule networks (Sabour et al., 2017) for zero-shot intent classification by transferring the prediction vectors from seen classes to unseen classes. ReCapsNet (Liu et al., 2019) shows that IntentCapsNet-ZS hardly recognizes utterances from unseen intents in the generalized zero-shot classification scenario, and proposes to solve this issue by transferring the transformation matrices from the seen to unseen intents. These approaches also need unknown intent embedding for classifying these instances. Our work do not require the assumption of that classes belong to a closed word. We don’t need the unseen intent samples to get the deep learning classifier to detect unknown intents as well.

3 Methodology

We train two different deep learning model for intent classification and use our post-processing steps on top of these to obtain optimal results. The pipeline of the system processes is shown in Figure 1. We describe the models along with our novel post-processing steps in this section.

3.1 Models

3.1.1 Bi-LSTM

We train Bi-directional Long Short Term Memory (Bi-LSTM) to obtain the prediction scores and use these scores to obtain the optimal thresholds for each known intent class using different threshold tuning methods as discussed in 3.3. Given an utterance with maximum class-ID length l, we transform a sequence of input words \(w_{1:l}\) into m-dimensional word embedding \(v_{1:l}\), which is used by forward and backward LSTM to produce feature representations x:

\[
\begin{align*}
\bar{x}_{l} &= LST M(v_{l}, \bar{c}_{l-1}) \\
\bar{x}_{l} &= LST M(v_{l}, \bar{c}_{l-1}) \\
x &= [\bar{x}_{l} : \bar{x}_{l}]
\end{align*}
\]

where \(v_{l}\) denotes the word embedding of input at time step \(l\). \(\bar{x}_{l}\) and \(\bar{x}_{l}\) are the output vector of forward and backward LSTM, respectively. \(\bar{c}_{l}\) and \(\bar{c}_{l}\) are the cell state vectors of forward and backward LSTM, respectively. We concatenate the last output vector of forward LSTM \(\bar{x}_{l}\) and the first output vector of backward LSTM \(\bar{x}_{l}\) into x as the sentence...
representation. It captures high-level semantic concepts learned by the model. The representation $x$ is then fed to an $n$ neuron feed forward layer where $n$ is the number of known intent classes in the dataset. The $n$ dimensional representation obtained is converted to probability distribution by using a ‘Softmax’ function.

3.1.2 BERT

We fine tune the pre-trained Bi-directional Encoder Representation from Transformer (BERT) model to get the ‘softmax’ classification scores of the input samples. Given $i^{th}$ input sentence $s_i$ we append a $[CLS]$ token at the beginning of the sentence. We obtain the token embeddings of the sequence $[CLS, T1, \ldots, TN] \in \mathbb{R}^{(N+1)\times H}$ from the last hidden layer of BERT. Here the $[CLS]$ vector representation is used for text classification, $N$ is the sequence length and $H$ is the hidden layer size. We calculate the prediction scores by applying ‘Softmax’ function to the last layer output (logits($x_i$)) of the trained BERT model.

3.2 Pre-Training

After training the intent classification model, for each input we obtain the ‘Softmax’ scores w.r.t each class at the output layer. We need to set a thresholds for these scores, above which the input sample is classified to the respective class. Since we do not use have any separate class for unknown intent, we train our model on a subset of the classes in the dataset, holding out the rest to be classified as unknown during testing. In order to reflect the effectiveness of the learned optimal thresholds we use cross-entropy loss $L_s$ to train our both the base models.

$$L_s = -\frac{1}{N} \sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

where $N$ is total number of training samples $y_i$ is true label and $\hat{y}_i$ is predicted label. Then, we use the pre-trained model to obtain the prediction scores of the input samples. These scores are used further for threshold tuning of each known intent class.

3.3 Finding Optimal Threshold for Each Known Intent Class

To obtain the prediction scores corresponding to each sample we pass the training data samples to the pre-trained classifiers. After getting the prediction scores we apply two different techniques to obtain the optimal threshold for each known intent class, viz. normal thresholding, and multi-objective optimization.

3.3.1 Normal Thresholding

In this method, first the input text containing the training data samples is fed to the deep learning classifier to get the prediction scores corresponding to each samples. These prediction scores ($PS$) and the list of thresholds ($T$) ranging from 0.1 to 0.9 increasing by 0.1 in each step is used to calculate the correct classification matrix ($CCM$) and the mis-classification matrix ($MCM$).

The set of prediction scores is a matrix of $N \times M$ where $N$ is the total number of training samples and $M$ is the number of known intent classes. This
set of prediction scores and the list of thresholds containing $K$ threshold values is used to calculate correct classification matrix ($CCM$) and the misclassification matrix ($MCM$).

Let $C(X)$ be the output class, $Y$ the ground truth class, and $\{,\}$ the enumeration function, the standard definition for correctly classified sample (or true positives) rate of an intent class $i$ is given in Equation 1:

$$\text{CC}_i = \frac{(C(X) = i \text{ AND } Y = i)}{Y = i} \quad (1)$$

We can also write the standard definition of misclassified sample rate (or false negatives) of an intent class $i$ as given by Equation 2:

$$\text{MC}_i = \frac{(C(X) \neq i \text{ AND } Y = i)}{Y = i} \quad (2)$$

The correct classification rate ($CC$) and misclassification (MC) rate of an intent $i$ can be extended by introducing the thresholds $\tau_i$ and by adding the unsure classification (UC) rate, for each intent as shown in Equation 3, 4 and 5.

$$\text{CC}_i(\tau_i) = \frac{(C(X) = i \text{ AND } S(X) > \tau_i) \text{ AND } (Y = i)}{Y = i} \quad (3)$$

$$\text{MC}_i(\tau_i) = \frac{(X) \neq i \text{ AND } S(X) > \tau_i) \text{ AND } (Y = i)}{Y = i} \quad (4)$$

$$\text{UC}_i(\tau_i) = \frac{((C(X) = i) \text{ or } (C(X) \neq i) \text{ AND } (S(X) < \tau_i) \text{ AND } (Y = i)}{Y = i} \quad (5)$$

For each intent we have:

$$\text{CC}_i(\tau) + \text{MC}_i(\tau) + \text{UC}_i(\tau) = 1$$

$CCM$ is a matrix of $K \times M$ dimension containing the correct classification rate of each intent class corresponding to each threshold in the threshold list i.e each entry $CC_{ij}$ is calculated using equation 6.

$$CC_{ij} = \frac{\sum_{i=1}^{N} (C(X) = i \text{ AND } S(X) > \tau_j) \text{ AND } (Y = i)}{Y = i} \quad (6)$$

$MCM$ is a matrix of $K \times M$ dimension containing the mis-classification rate of each intent class corresponding to each threshold in the threshold list i.e each entry $MC_{ij}$ is calculated using equation 7.

$$MC_{ij} = \frac{\sum_{i=1}^{N} (C(X) \neq i \text{ AND } S(X) > \tau_j) \text{ AND } (Y = i)}{Y = i} \quad (7)$$

After obtaining these two matrices, we obtain optimal $\tau_j$ for each known intent class by the following technique. We keep the best correct classification rate while reducing the mis-classification rate. For this, we use two steps. First, we determine the threshold(s) $\tau$ which maximizes $CC_i(\tau)$. Since several thresholds could reach this maximum, we obtain a set of threshold(s) $Seg_1$. Then, we selected the threshold with the lower $MC_i(\tau)$. This can be mathematically written as:

$$s = \argmax_{\tau} (CC_i(\tau))$$

$$\tau_i = \argmin_{\tau \in Seg_1} (MC_i(\tau))$$

3.3.2 Multi-Objective Optimization (NSGA2)

To get the optimal threshold we use Non-dominated Sorting Genetic Algorithm II (NSGA-II) which...
is a multi-objective genetic algorithm, proposed by (Deb et al., 2002). In the structure of NSGA-II, in addition to genetic operators, crossover and mutation, two specialized multi-objective operators and mechanisms are defined and utilized. These are as follows:

- **Non-Dominated Sorting**: The population is sorted and partitioned into fronts (F1, F2, etc.), where F1 (first front) indicates the approximated Pareto front.

- **Crowding Distance**: It is a mechanism of ranking among members of a front, which are dominating or dominated by each other.

We optimize for two objective (i). Correct classification rate \((CC)\), and (ii). Precision of the known intents. The NSGA2 takes threshold values of an intent as the input variable (values ranging from 0.1 to 0.99). It then uses prediction scores of samples from the pre-trained base model to perform optimization on the two objective functions, explained in details in Section 3.3.1 to get an optimal threshold for each known intent class. We initialize the population by randomly selecting the values from the range of the threshold variables and then we calculate the two objective values for each entry in the initial population.

Next we perform a non-dominated sorting in the combination of parent and offspring populations and classify them by fronts, i.e. these are sorted in an ascending level of non-domination. Next, we fill new population according to front ranking. If one front is taken partially, crowding-sort is performed. The less dense population are preferred. The offspring population (children) is then created from this new population using crowded tournament selection (It compares by front ranking, if equal then by crowding distance), crossover and mutation operators. The most important solutions (i.e. the best entries) of the population are kept in fronts.

We run the same procedure 1000 times to get a set of optimal thresholds for each known intent class. From this set of thresholds we choose the maximum threshold. This optimal threshold is used to decide upon known and unknown intent samples.

### 3.4 Testing

During testing, when a new sample (unseen class) is encountered it is first fed to the base model (BiLSTM or BERT) to get the corresponding prediction scores. After getting the prediction scores we compare each entries in the prediction scores with the corresponding optimal thresholds and if we find all the entries to be less than the corresponding optimal thresholds we classify that sample as unknown else we classify the sample to the one known intent class for which the prediction score is higher than the corresponding optimal threshold.

## 4 Datasets and Experiments

### 4.1 Dataset

We use three datasets to conduct our experiments. The detailed statistics of the datasets are shown in Table 1. Few example intents from each dataset are shown in Table 2.

#### 4.1.1 Banking

This dataset contains fine-grained intents in the banking domain (Casanueva et al., 2020). It contains 77 intents and 13,083 customer service queries.

#### 4.1.2 Bank-Catridge

This is a real-world banking domain chat dataset which contains manually updated samples, created through paraphrasing followed by manual verification. This dataset consists of 14 intents in total, consisting of almost 100 samples per intent.

#### 4.1.3 SNIPS-NLU

SNIPS-NLU is an English natural language corpus collected in a crowd-sourced fashion to benchmark the performance of voice assistants. It contains 7 intents and almost 2000 samples per intent.

### 4.2 Experimental Setups

We keep 25% of the overall intent classes in training and validation set as masked while keeping these masked intent samples in the test set as unmasked. To have a fair evaluation on the imbalanced dataset, we randomly select known classes by weighted random sampling without replacement in the training and validation sets. For BERT initialization, we use the weights of the ‘bert-base-uncased’ model containing 8-layers transformer units. We fine-tune the model on our training sets. We keep learning-rate to 5e-5, the training batch size is 64 and number of training epochs is set to 50. For Bi-LSTM, we set the output dimension as 128 upon which final linear layer is built (according to the number of classes in the dataset). The maximum number of epochs is set to 50 with early
Table 3: Samples texts whose intents are mis-classified by the ADB model but are correctly identified by our BERT+NSGA2 model

<table>
<thead>
<tr>
<th>Text</th>
<th>True Label</th>
<th>Predicted Label (ADB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is Visa or Mastercard available?</td>
<td>visa or mastercard</td>
<td>supported cards</td>
</tr>
<tr>
<td>The app is showing an ATM withdrawal that I didn’t make.</td>
<td>cash withdrawal</td>
<td>declined cash withdrawal</td>
</tr>
<tr>
<td>I did what you told me earlier and contacted the seller for a refund directly, but nothing is happening! It’s been a week and I still haven’t got anything. Please just give me back my money.</td>
<td>refund not showing</td>
<td>balance not updated after cheque or cash deposit</td>
</tr>
</tbody>
</table>

Table 4: The F1 scores of detecting unknown intent class samples with 75% of total intent class as known class on BANKING, Bank Catridge and SNIPS dataset.

<table>
<thead>
<tr>
<th></th>
<th>BANKING</th>
<th>Bank Catridge</th>
<th>SNIPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADB</td>
<td>66.47</td>
<td>72.1</td>
<td>69</td>
</tr>
<tr>
<td>Bi-LSTM + NT</td>
<td>22.15</td>
<td>64.2</td>
<td>33.5</td>
</tr>
<tr>
<td>Bi-LSTM + NSGA2</td>
<td>35.2</td>
<td>82.1</td>
<td>49</td>
</tr>
<tr>
<td>BERT+NT</td>
<td>67.2</td>
<td>66.1</td>
<td>54.3</td>
</tr>
<tr>
<td>BERT+NSGA2</td>
<td>67.2</td>
<td>75</td>
<td>90.1</td>
</tr>
</tbody>
</table>

5 Result and Analysis

We experiment with different variants of the proposed model as follows: (i). Bi-LSTM + normal-thresholding, (ii). Bi-LSTM + NSGA2 and (iii). BERT + normal-thresholding and (iv). BERT+NSGA2. We also re-implement the ADM model (Shu et al., 2017) and obtain the results on the datasets mentioned in Section 4. Table 4 shows the F1 score of detecting unknown intent class samples with 75% of total intent class being kept as known on Banking, Bank Catridge and SNIPS-NLU dataset. The best results are highlighted in bold. Comparing with the best score of baseline and different variants of our approach we can see that our final model BERT+NSGA2 gives better results than the baseline and the different variants of our proposed model. Comparing with ADB our approach yields 0.7% improvement on Banking dataset, 3% improvement on Bank catridge dataset, and 21% improvement on SNIPS dataset. It can be observed by the results that our BERT+NSGA2 based approach is able to learn tight thresholds to clearly distinguish between known and unknown intent samples. Using Normal thresholding technique where the objective functions are optimised sequentially does not work well as optimizing one objective function can counter the optimization of another objective. This problem is addressed by multi-objective optimization based technique which simultaneously satisfies all the objective functions, finds a set of optimal solutions instead of one optimal solution. Some examples that are correctly classified by the BERT+NSGA2 and not by BERT+NT are shown in Table 5. We can see that multi-objective optimization plays a vital role in predicting the unknown samples correctly as compared to normal optimization.

Some examples that are correctly classified by the BERT+NSGA2, but not by ADB are shown in Table 3. From the examples we observe that our BERT+NSGA2 gives importance to the words which are there in the unknown intent samples like “refund”, “visa”, “master_card” and “didn’t make” to make the decision between known and unknown intent class. On the other hand, Table 6 shows that there are some samples in the test data which can be miss-classified to one of the similar intent classes. For example the text “I transferred my funds, why did it not go through?” can be miss-classified to “declined_transfer” intent but it actually belongs...
What is the number of days I have to wait for my Europe transfer?
I need to find out why my transfer didn’t get there.
I have a pending cash withdrawal
I don’t find your services useful anymore, how do I delete my account?
Will it cost more money if my currency needs to be exchanged?

<table>
<thead>
<tr>
<th>Text</th>
<th>True Label</th>
<th>Predicted Label (BERT+NT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the number of days I have to wait for my Europe transfer?</td>
<td>balance not updated after bank transfer</td>
<td>transfer timing</td>
</tr>
<tr>
<td>I need to find out why my transfer didn’t get there.</td>
<td>declined transfer</td>
<td>transfer not received by recipient</td>
</tr>
<tr>
<td>I have a pending cash withdrawal</td>
<td>balance not updated after cheque or cash deposit</td>
<td>pending cash withdrawal</td>
</tr>
<tr>
<td>I don’t find your services useful anymore, how do I delete my account?</td>
<td>edit personal details</td>
<td>terminate account</td>
</tr>
<tr>
<td>Will it cost more money if my currency needs to be exchanged?</td>
<td>exchange via app</td>
<td>exchange charge</td>
</tr>
</tbody>
</table>

Table 5: Samples texts whose intents are mis-classified by the BERT + NT model but are correctly identified by out BERT + NSGA2 model

<table>
<thead>
<tr>
<th>text</th>
<th>true_intent</th>
<th>predicted_intent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where can I exchange my money for EUR?</td>
<td>fiat_currency_support</td>
<td>exchange_via_app</td>
</tr>
<tr>
<td>I transferred money yesterday, but it still isn’t available?</td>
<td>pending_transfer</td>
<td>balance_not_updated_after_bank_transfer</td>
</tr>
<tr>
<td>I transferred my funds, why did it not go through?</td>
<td>failed_transfer</td>
<td>declined_transfer</td>
</tr>
<tr>
<td>My card still hasn’t arrived after 2 weeks. Is it lost?</td>
<td>card_arrival</td>
<td>lost_or_stolen_card</td>
</tr>
<tr>
<td>How can I fund my top-up account using my bank account?</td>
<td>transfer_into_account</td>
<td>topping_up_by_card</td>
</tr>
</tbody>
</table>

Table 6: Samples texts which can be miss-classified to a very similar intent.

Acknowledgement

The research reported in this paper is supported by the project “Autonomous Goal-Oriented and Knowledge-Driven Neural Conversational Agents”, sponsored by Accenture LLP. Prerna acknowledges Accenture LLP for the research internship.

References


Are the Multilingual Models Better?
Improving Czech Sentiment with Transformers

Pavel Přibáň1,2 and Josef Steinberger1,2

University of West Bohemia, Faculty of Applied Sciences, Czech Republic
1NTIS – New Technologies for the Information Society,
2Department of Computer Science and Engineering,
{pribanp,jstein}@kiv.zcu.cz
http://nlp.kiv.zcu.cz

Abstract

In this paper, we aim at improving Czech sentiment with transformer-based models and their multilingual versions. More concretely, we study the task of polarity detection for the Czech language on three sentiment polarity datasets. We fine-tune and perform experiments with five multilingual and three monolingual models. We compare the monolingual and multilingual models’ performance, including comparison with the older approach based on recurrent neural networks. Furthermore, we test the multilingual models and their ability to transfer knowledge from English to Czech (and vice versa) with zero-shot cross-lingual classification. Our experiments show that the huge multilingual models can overcome the performance of the monolingual models. They are also able to detect polarity in another language without any training data, with performance not worse than 4.4% compared to state-of-the-art monolingual trained models. Moreover, we achieved new state-of-the-art results on all three datasets.

1 Introduction

In recent years, BERT-like models (Devlin et al., 2019) based on the Transformer architecture (Vaswani et al., 2017) and generalized language models brought a significant improvement in performance in almost any NLP task (Raffel et al., 2020a), especially in English. Despite this fact, not much work has been recently done in sentiment analysis for the Czech language with the latest Transformer models. We partly fill this gap by focusing on the Sentiment Classification task, also known as Polarity Detection.

Polarity detection is a classification task where the goal is to assign a sentiment polarity to a given text. The positive, negative and neutral classes are usually used as the polarity labels. The polarity can also be defined with a different number of labels, i.e., fine-grained sentiment analysis (Liu, 2012).

The models based on BERT were almost exclusively trained for English, limiting their usage to other languages. Recently, however, their cross-lingual adaptions like mBERT (Devlin et al., 2019), mT5 (Xue et al., 2020), XLM (Conneau and Lample, 2019) or XLM-R (Conneau et al., 2020) emerged along with other non-English monolingual versions, for example, Czech (Sido et al., 2021), French (Martin et al., 2020; Le et al., 2019), Arabic (Safaya et al., 2020), Romanian (Dumitrescu et al., 2020), Dutch (Vries et al., 2019) or Finnish (Virtanen et al., 2019).

Our motivation is to reveal the performance limits of the current SotA transformer-based models on the Czech polarity detection task, check the ability of the multilingual models to transfer knowledge between languages and unify the procedure and data that enable the correct future evaluation of this task.

In this paper, we focus on the task of polarity detection applied on Czech text by comparing the performance of seven pre-trained transformer-based models (both monolingual and multilingual) on three Czech datasets. We fine-tune each model on each dataset and we provide a comprehensive survey of their performance. Our experiments show the effectiveness of the Transformer models that significantly outperform the older approaches based on recurrent neural networks. We observe that the monolingual models can be notably outperformed by the multilingual models, but only by those with much more parameters. Moreover, we achieve new state-of-the-art results on all three evaluated datasets.

We are also interested in the ability of the multilingual models to transfer knowledge between languages and its usability for polarity detection. Thus, we perform zero-shot cross-lingual classification,
fine-tune four cross-lingual transformer-based models on the English dataset and then test the models on Czech data. We also perform the same experiment in the reverse direction, i.e., from Czech to English. The results reveal that the XLM-R-Large model (fine-tuned solely on English) can achieve very competitive results that are only about 4 % worse than the SotA model fine-tuned by us on Czech data. To the best of our knowledge, this is the first paper that performs zero-shot cross-lingual polarity detection for the Czech language.

We also noticed that the comparison with the previous works is rather problematic and thus, we provide a split for all Czech datasets that allows comparing future works much easier. Our code and pre-trained models are publicly available1.

Our main contributions are the following: 1) We provide the comprehensive performance comparison of the currently available transformer-based models for the Czech language on the polarity detection task along with the models’ optimal settings. 2) We test the ability of the multilingual models to transfer knowledge between Czech and English. 3) We release all the fine-tuned models and code freely for research purposes and we provide a data split that allows future comparison and evaluation. Furthermore, we achieved new state-of-the-art results for all three evaluated datasets.

2 Related Work

The previous approaches (Kim, 2014; Johnson and Zhang, 2016; Cliche, 2017; Baziotis et al., 2017; Gray et al., 2017; Conneau et al., 2017) for English polarity detection and other related tasks mainly relied on transfer learning and pre-trained word embeddings such as word2vec (Mikolov et al., 2013) and fastText (Bojanowski et al., 2017) in combinations with Convolutional Neural Networks (CNN) or Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), eventually in conjunction with the modified attention mechanism (Bahdanau et al., 2015; Rocktäschel et al., 2015; Raffel and Ellis, 2015). Furthermore, the new contextualized word representations such as CoVe (McCann et al., 2017) or ELMo (Peters et al., 2018) and pre-trained language model ULMFiT (Howard and Ruder, 2018) were successfully applied to the polarity detection. Finally, the latest transformer-based models like BERT (Devlin et al., 2019), GPT (Radford et al., 2018), RoBERTa (Liu et al., 2019) or T5 (Raffel et al., 2020b) that are all in general trained on language modeling tasks proved their performance superiority for English over all previous approaches, for example in (Sun et al., 2019). These models are pre-trained on a modified language modeling tasks with a huge amount of unlabeled data. In the end, they are fine-tuned for a specific downstream task.

The initial works on Czech polarity detection and sentiment analysis usually used lexical features (Steinberger et al., 2011; Veselovská et al., 2012) or Bag-of-Words text representations along with the Naive Bayes or logistic regression classifiers (Habernal et al., 2013) or a combination of supervised and unsupervised approach (Brychcin and Habernal, 2013). Lenc and Hercig (2016) applied CNN using the architecture from (Kim, 2014) and the LSTM neural network to all three datasets that we use in this paper. Another usage of LSTM neural network with the self-attention mechanism (Humphreys and Sui, 2016) can be found in (Libovický et al., 2018). Similarly, Sido and Konopič (2019) tried to use curriculum learning with CNN and LSTM.

Lehečka et al. (2020) pre-trained a BERT-based model for polarity detection with an improved pooling layer and distillation of knowledge technique. The most recent application of the Transformer model is in (Sido et al., 2021). The authors created a BERT model for Czech and, as one of the evaluation tasks, they performed polarity detection on the FB and CSFD datasets.

To the best of our knowledge, there are no previous works that focus on the zero-shot cross-lingual polarity detection task in the Czech language. The recent related work can be found in (Eriguchi et al., 2018), where the authors use the neural machine translation encoder-based model and English data to perform zero-shot cross-lingual sentiment classification on French. In (Eriguchi et al., 2018) the authors performed the zero-shot classification from Slovene to Croatian. Another related work can be found in (Wang and Banko, 2021; Qin et al., 2020).

3 Data

To the best of our knowledge, there are three Czech publicly available datasets for the polarity detection task: (1) movie review dataset (CSFD), (2) Facebook dataset (FB) and (3) product review dataset (Mallcz), all of them come from (Habernal et al.,

---

1https://github.com/pauli31/improving-czech-sentiment-transformers
and each text sample is annotated with one of three labels, i.e., positive, neutral and negative, see Table 1 for the class distribution. For the cross-lingual experiments we use the two-class English movie review dataset (IMDB) (Maas et al., 2011).

### Table 1: Datasets statistics.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Part</th>
<th>Positive</th>
<th>Negative</th>
<th>Neutral</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSFD</td>
<td>train</td>
<td>22 117</td>
<td>21 441</td>
<td>22 235</td>
<td>65 793</td>
</tr>
<tr>
<td></td>
<td>dev</td>
<td>2 456</td>
<td>2 399</td>
<td>2 456</td>
<td>7 311</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>6 324</td>
<td>5 876</td>
<td>6 077</td>
<td>18 277</td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>30 877</td>
<td>30 616</td>
<td>30 768</td>
<td>91 321</td>
</tr>
<tr>
<td>FB</td>
<td>train</td>
<td>1 605</td>
<td>1 227</td>
<td>3 311</td>
<td>6 143</td>
</tr>
<tr>
<td></td>
<td>dev</td>
<td>1 151</td>
<td>1 513</td>
<td>2 926</td>
<td>5 590</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>8 11</td>
<td>6 13</td>
<td>1 502</td>
<td>2 926</td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>2 587</td>
<td>1 991</td>
<td>5 174</td>
<td>9 752</td>
</tr>
<tr>
<td>Mallcz</td>
<td>train</td>
<td>7 410</td>
<td>7 498</td>
<td>23 022</td>
<td>104 620</td>
</tr>
<tr>
<td></td>
<td>dev</td>
<td>8 253</td>
<td>8 48</td>
<td>2 324</td>
<td>11 652</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>20 624</td>
<td>2 043</td>
<td>6 977</td>
<td>29 062</td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>31 287</td>
<td>10 668</td>
<td>31 353</td>
<td>73 943</td>
</tr>
<tr>
<td>IMDB</td>
<td>train</td>
<td>12 500</td>
<td>12 500</td>
<td>-</td>
<td>25 000</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>12 500</td>
<td>12 500</td>
<td>-</td>
<td>25 000</td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>25 000</td>
<td>25 000</td>
<td>-</td>
<td>50 000</td>
</tr>
</tbody>
</table>

The FB dataset contains 10k random posts from nine different Facebook pages that were manually annotated by two annotators. The CSFD dataset is created from 90k Czech movie reviews from the Czech movie database\(^2\) that were downloaded and annotated according to their star rating (0–2 stars as negative, 3–4 stars as neutral, 5–6 stars as positive). The Mallcz dataset consists of 145k users’ reviews of products from Czech e-shop\(^3\), the labels are assigned according to the review star rating on the scale 0-5, where the reviews with 0-3 stars are labeled as negative, 4 stars as neutral and 5 stars as positive. The English IMDB dataset includes 50k movie reviews scraped from the Internet Movie Database\(^4\) with positive and negative classes split into training and testing parts of equal size.

Since there is no official partitioning for the Czech datasets, we split them into training, development and testing parts with the same class distribution for each part as it is in the original dataset, see Table 1. For the Mallcz and CSFD datasets, we use the following ratio: 80% for training data, 20% for testing data, for the FB dataset, it is 70% and 30%, respectively and 10% from the training data (for all datasets) is used as the development data. We used different split ratio for the FB dataset because it is approximately ten and sixteen times smaller than the CSFD and Mallcz datasets, respectively and we did not want to reduce the size of the testing data too much.

### 4 Models Description

We performed exhaustive experiments with transformed-based models and in order to compare them with the previous works, we also implemented the older models (baseline models) that include the logistic regression classifier and the BiLSTM neural network.

#### 4.1 Baseline Models

We re-implemented the best models from (Habernal et al., 2013), i.e., logistic regression classifier (lrc) with character n-grams (in a range from 3-grams to 6-grams), word uni-grams and bi-grams features. The second baseline model is the LSTM model partly inspired by (Baziotis et al., 2017). Its input is a sequence of \(t\) tokens represented as a matrix \(M \in \mathbb{R}^{t \times d}\), where \(d = 300\) is a dimension of the Czech pre-trained fastText word embeddings (Bojanowski et al., 2017)\(^5\). The maximal size of the input vocabulary is set to 300 000. The input is passed into the trainable embedding layer that is followed by two BiLSTM (Graves and Schmidhuber, 2005) layers and after each, the dropout (Srivastava et al., 2014) is applied. After the two BiLSTM layers, the self-attention mechanism is applied. The output is then passed to a fully-connected softmax layer. An output of the softmax layer is a probability distribution over the possible classes. We use the Adam (Kingma and Ba, 2014) optimizer with default parameters \((\beta_1 = 0.9, \beta_2 = 0.999)\) and with weight decay modification (Loshchilov and Hutter, 2017) and the cross-entropy loss function. We replace numbers, emails and links with generic tokens, we tokenize input text with the TokTok tokenizer\(^6\) and we use a customized stemmer\(^7\).

We use different hyper-parameters for each dataset, see Appendix A.1 for the complete settings.

---

\(^2\)The FB dataset also contains 248 samples with a fourth class called bipolar, but we ignore this one.

\(^3\)Available at \(https://fasttext.cc/docs/en/crawl-vectors.html\)

\(^4\)https://www.mall.cz

\(^5\)https://www.imdb.com

\(^6\)https://github.com/jonsafari/tok-tok

\(^7\)https://github.com/UFAL-DSG/alex/blob/master/alex/utils/czech_stemmer.py


### 4.2 Transformer Models

In total, we use eight different transformer-based models (five of them are multilingual). All of them are based on the original BERT model. They use only the encoder part of the original Transformer (Vaswani et al., 2017), although their pre-training procedure may differ. There are also text-to-text models like T5 (Raffel et al., 2020b) and BART (Lewis et al., 2019) and their multilingual versions mT5 (Xue et al., 2020) and mBART (Liu et al., 2020; Tang et al., 2020). The main difference from BERT-like models is that they use the full encoder-decoder architecture of the Transformer. They are mainly intended for text generation tasks (e.g., abstractive summarization). We decided to use only the BERT-like models with the same architecture because they fit more for the classification task.

All the models are pre-trained on a modified language modeling task, for example, Masked Language Modeling (MLM) and eventually on some classification task like Next Sentence Prediction (NSP) or Sentence Ordering Prediction (SOP), see (Devlin et al., 2019; Lan et al., 2020) for details. The evaluated models differ in the number of parameters (see Table 2) and thus, their performance is also very different, see Section 5.

<table>
<thead>
<tr>
<th>Model</th>
<th>#Params</th>
<th>Vocab</th>
<th>#Langs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czert-B</td>
<td>110M</td>
<td>30k</td>
<td>1</td>
</tr>
<tr>
<td>Czert-A</td>
<td>12M</td>
<td>30k</td>
<td>1</td>
</tr>
<tr>
<td>RandomALBERT</td>
<td>12M</td>
<td>30k</td>
<td>1</td>
</tr>
<tr>
<td>mBERT</td>
<td>177M</td>
<td>120k</td>
<td>104</td>
</tr>
<tr>
<td>SlavicBERT</td>
<td>177M</td>
<td>120k</td>
<td>4</td>
</tr>
<tr>
<td>XLM</td>
<td>570M</td>
<td>200k</td>
<td>100</td>
</tr>
<tr>
<td>XLM-R-Base</td>
<td>270M</td>
<td>250k</td>
<td>100</td>
</tr>
<tr>
<td>XLM-R-Large</td>
<td>559M</td>
<td>250k</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 2: Models statistics with a number of parameters, vocabulary size and a number of supported languages.

**Czert-B** is the Czech version of the of the original BERTBASE model (Devlin et al., 2019). The only difference is that during the pre-training, the authors increased the batch size to 2048 and they slightly modified the NSP prediction task (Sido et al., 2021).

**Czert-A** is the Czech version of the ALBERT model (Lan et al., 2020), also with the same modification as Czert-B, i.e., batch size was set to 2048 and the modified NSP prediction task is used instead of the SOP task (Sido et al., 2021).

**RandomALBERT** we follow the evaluation in (Sido et al., 2021) and we also test randomly initialized ALBERT model without any pre-training to show the importance of pre-training of such models and its performance influence on the polarity detection task.

**mBERT** (Devlin et al., 2019) is a multilingual version of the original BERTBASE, jointly trained on 104 languages.

**SlavicBERT** (Arkhipov et al., 2019) is initialized from the mBERT checkpoint and further pre-trained with a modified vocabulary only for four Slavic languages (Bulgarian, Czech, Polish and Russian).

**XLM** (Conneau and Lample, 2019) utilizes the training procedure of the original BERT model for multilingual settings mainly by using the Byte-Pair Encoding (BPE) and increasing the shared vocabulary between languages.

**XLM-R-Base** (Conneau et al., 2020) is a multilingual version of the RoBERTas (Liu et al., 2019) specifically optimized and pre-trained for 100 languages.

**XLM-R-Large** (Conneau et al., 2020) is the same model as the XLM-R-Base, but it is larger (it has more parameters).

### 4.3 Transformers Fine-Tuning

To utilize the models for text classification, we follow the default approaches mentioned in the corresponding models’ papers and we fine-tune all parameters of the models. In all models except XLM, we use the final hidden vector \( \mathbf{h} \in \mathbb{R}^H \) of the special classification token \([CLS]\) or \(<s>\) taken from the pooling layer\(^9\) of BERT or RoBERTa models, respectively. The vector \( \mathbf{h} \) represents the entire encoded sequence input, where \( H \) is the hidden size of the corresponding model. We add a task-specific linear layer (with a dropout set to 0.1) represented by a matrix \( \mathbf{W} \in \mathbb{R}^{C \times H} \), where \( C \) is a set of classes. We compute the classification output, i.e., the input sample being classified as class \( c \in C \) as \( c = \arg \max (\mathbf{hW}^T) \).

In the case of the XLM model, we take the last hidden state (without any pooling layer) of the first input token and we apply the same linear layer \( \mathbf{W} \) and approach to obtain the classification output. For learning, we use the Adam optimizer with default parameters and with weight decay (same as for the \textsc{LSTM} model), and the cross-
entropy loss function. See Section 5.1 and Appendix A.2 for the hyper-parameters we used.

5 Experiments & Results

We perform two types of experiments, i.e., monolingual and cross-lingual. In monolingual experiments, we fine-tune and evaluate the Transformer models for each dataset separately on three-class (positive, negative, and neutral) and two-class (positive and negative) sentiment analysis. We also implemented the logistic regression (lrc) and LSTM baseline models and we compare the results with the existing works.

In cross-lingual experiments, we test the ability of four multilingual transformer-based models to transfer knowledge between English and Czech. We run the multilingual models only on the two-class datasets (positive and negative). We fine-tune either on English (IMDB) or Czech (CSFD), and then we evaluate on the other language. Thus we perform the zero-shot cross-lingual classification. We decided to use the IMDB and CSFD dataset because they are from the same domain i.e., movie reviews.

Each experiment\(^\text{10}\) was performed at least five times and we report the results using the macro \(F_1\) score.

5.1 Monolingual Experiments

The goal of the monolingual experiments is to reveal the current state-of-the-art performance on the Czech polarity datasets, namely CSFD, FB and Malicz (see Section 3) and provide a comparison between the available models and their settings.

As we already mentioned, we split the datasets into training, development and testing parts. There is no official split for the datasets and we found out that all the available works usually use either 10-fold cross-validation or they split\(^\text{11}\) the datasets on their own, the † and * symbols in Table 3, respectively causing the comparison to be difficult.

We fine-tune all models on training data and we measure the results on the development data. We select the model with the best performance on the development data and we fine-tune the model on combined training and development data. We report the results in Table 3 on the testing data with 95% confidence intervals.

Firstly, we re-implemented the logistic regression classifier (lrc) with the best feature combination from (Habernal et al., 2013) and we report the results on our data split. We can see that we obtained very similar results to the ones stated in (Habernal et al., 2013). We also tried to improve this baseline with TF-idf weighting, but it did not lead to any significant improvements, so we decided to keep the settings the same as in (Habernal et al., 2013), so the results are comparable.

For the LSTM model, we tried different combinations of hyper-parameters (learning rate, optimizer, dropout, etc.). We report the used hyper-parameters for the results from Table 3 in Appendix A.2. Our implementation is only about 1 % worse than LSTM with the self-attention model from (Libovický et al., 2018), but they used a different data split. For the Mallcz dataset, we were not able to outperform the lrc baseline with the LSTM model.

We fine-tune all parameters of the seven pre-trained BERT-based models and one randomly initialized ALBERT model. In our experiments, we use constant learning rate and also linear learning rate decay (without learning rate warm-up) with the following initial learning rates: 2e-6, 2e-5 and 2.5e-5. We got inspired by the ones used in (Sun et al., 2019). Based on the average number of tokens for each dataset and models’ tokenizer (see Table 4 and Figures 1, 2, 3)\(^\text{12}\), we use a max sequence length of 64 and a batch size of 32 for the FB dataset. We restrict the max sequence length for the CSFD and Mallcz datasets to 512 and use a batch size of 32. All other hyper-parameters of the models are set to the pre-trained models’ defaults. See Table 7 in Appendix A.2 for the reported results’ hyper-parameters.

We repeated all the basic experiments with the polarity detection task from (Sido et al., 2021) with the new data split. Our results do not significantly differ, as shown in Table 8 and in Appendix A.2. If we compare the BERT model from (Lehečka et al., 2020) with the Czert-B, mBERT and SlavicBERT models\(^\text{13}\), we can see that on the binary task, they also perform very similarly, i.e., around 93 %, but again they used different test data (the entire CSFD dataset\(^\text{14}\)). The obvious observation is that the XLM-R-Large model is supe-

---

\(^{10}\) Except for the experiments with the lrc model.

\(^{11}\) The authors do not provide any recipe to reproduce the results.

\(^{12}\) The distributions of the other models were similar to those shown in the mentioned Figures.

\(^{13}\) All of them should have the same or almost the same architecture and a similar number of parameters.

\(^{14}\) The examples with positive and negative classes.
<table>
<thead>
<tr>
<th>Model</th>
<th>CSFD</th>
<th>FB</th>
<th>Mallcz</th>
</tr>
</thead>
<tbody>
<tr>
<td>lrc (ours)</td>
<td>79.63 ± 0.18</td>
<td>67.86 ± 0.49</td>
<td>76.71 ± 0.24</td>
</tr>
<tr>
<td>LSTM (ours)</td>
<td>79.88 ± 0.10</td>
<td>72.89 ± 0.49</td>
<td>74.33 ± 0.12</td>
</tr>
<tr>
<td>Czert-A</td>
<td>79.80 ± 0.60</td>
<td>73.06 ± 0.50</td>
<td>76.79 ± 0.38</td>
</tr>
<tr>
<td>Czert-B</td>
<td>84.80 ± 0.10</td>
<td>76.90 ± 0.38</td>
<td>79.35 ± 0.24</td>
</tr>
<tr>
<td>mBERT</td>
<td>82.74 ± 0.16</td>
<td>71.61 ± 0.13</td>
<td>70.79 ± 5.74</td>
</tr>
<tr>
<td>SlavicBERT</td>
<td>82.59 ± 0.12</td>
<td>73.93 ± 0.53</td>
<td>75.34 ± 2.54</td>
</tr>
<tr>
<td>RandomALBERT</td>
<td>75.79 ± 0.18</td>
<td>62.53 ± 0.46</td>
<td>64.81 ± 0.25</td>
</tr>
<tr>
<td>XLM-R-Base</td>
<td>84.82 ± 0.10</td>
<td>77.81 ± 0.50</td>
<td>75.43 ± 0.07</td>
</tr>
<tr>
<td>XLM-R-Large</td>
<td>87.08 ± 0.11</td>
<td>81.70 ± 0.64</td>
<td>79.81 ± 0.21</td>
</tr>
<tr>
<td>XLM</td>
<td>83.67 ± 0.11</td>
<td>71.46 ± 1.58</td>
<td>77.56 ± 0.08</td>
</tr>
</tbody>
</table>

| (Libovický et al., 2018)* | 80.80 ± 2.54 | 85.38 ± 0.10 | 82.31 ± 0.07 |
| (Libovický et al., 2018)* | 80.80 ± 0.10 | -             | -             |
| (Lehečka et al., 2020)*   | 79.00 ± 1.58 | 79.81 ± 0.01 | 94.37 ± 0.02 |

Table 3: The final monolingual results as macro $F_1$ score for all three Czech polarity datasets on two and three classes. For experiments with neural networks performed by us, we present the results with a 95% confidence interval. The models from papers marked with † were evaluated with 10-fold cross-validation and the ones marked with * were evaluated on custom data split.

<table>
<thead>
<tr>
<th>Model</th>
<th>CSFD</th>
<th>FB</th>
<th>Mallcz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czert-B</td>
<td>84.5 ± 0.10</td>
<td>20.3 ± 0.64</td>
<td>34.3 ± 1.47</td>
</tr>
<tr>
<td>mBERT</td>
<td>111.6 ± 1.24</td>
<td>25.6 ± 0.66</td>
<td>46.6 ± 2.03</td>
</tr>
<tr>
<td>SlavicBERT</td>
<td>83.6 ± 0.98</td>
<td>20.7 ± 0.62</td>
<td>34.3 ± 1.41</td>
</tr>
<tr>
<td>XLM-R-Base</td>
<td>100.5 ± 1.08</td>
<td>22.6 ± 0.64</td>
<td>41.0 ± 1.81</td>
</tr>
<tr>
<td>XLM-R-Large</td>
<td>81.7 ± 0.99</td>
<td>19.7 ± 0.62</td>
<td>32.6 ± 1.43</td>
</tr>
<tr>
<td>XLM</td>
<td>93.9 ± 0.95</td>
<td>20.4 ± 0.53</td>
<td>37.5 ± 1.67</td>
</tr>
<tr>
<td>XLM-R-Fine</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: The average and maximum number of subword tokens for each model’s tokenizer and dataset.

5.2 Cross-lingual Experiments

The cross-lingual experiments were performed with the multilingual models that support English and Czech. For these experiments, we use linear learning rate decay with an initial learning rate of $2e^{-6}$.

Firstly, we fine-tuned the models on the English IMDB dataset and we evaluated them on the test part of the Czech binary CSFD dataset (i.e., zero-shot cross-lingual classification). We randomly selected 5k examples from the IMDB dataset as the development data. The rest of the 45k examples was used as training data. We select the models that perform best on the English development data\footnote{The model was trained for a maximum of 15 epochs and it would probably get better with a higher number of epochs, but the other models were trained for the same or lower number of epochs.} and we report the results in Table 5. The $\text{dev (en)}$ column refers to results obtained on the CSFD testing part. For easier comparison, we also include the Monoling. (cs) column that contains the results (same as in Table 3) for models trained on Czech data. The XLM-R-Large was able to achieve results only about 4.4% worse than the same model that was fine-tuned on Czech data. It is a great result if we consider that the model has never seen any labeled Czech data. The XLM and mBERT models perform much worse.

<table>
<thead>
<tr>
<th>Model</th>
<th>EN $\rightarrow$ CS</th>
<th>Monoling. (cs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dev (en)</td>
<td>test (cs)</td>
</tr>
<tr>
<td>XLM-R-Base</td>
<td>94.52 ± 0.12</td>
<td>88.01 ± 0.28</td>
</tr>
<tr>
<td>XLM-R-Large</td>
<td>95.86 ± 0.06</td>
<td>91.61 ± 0.06</td>
</tr>
<tr>
<td>XLM</td>
<td>92.76 ± 0.34</td>
<td>75.37 ± 0.29</td>
</tr>
<tr>
<td>mBERT</td>
<td>93.07 ± 0.03</td>
<td>76.32 ± 1.13</td>
</tr>
</tbody>
</table>

Table 5: Macro $F_1$ score for cross-lingual experiments from English to Czech.
the current state-of-the-art works (Thongtan and Phienthrakul, 2019; Sun et al., 2019) use this metric. Similarly to the previous case, we selected the model that performs best on Czech CSFD development data. For these experiments, the mBERT did not converge. As in the previous experiment, the XLM-R-Large performs best and it achieves almost 94% accuracy that is only 3.4% below the current SotA result from (Thongtan and Phienthrakul, 2019).

<table>
<thead>
<tr>
<th>Model</th>
<th>CS → EN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dev (cs)</td>
</tr>
<tr>
<td>XLM-R-Base</td>
<td>94.22 ± 0.01</td>
</tr>
<tr>
<td>XLM-R-Large</td>
<td>95.65 ± 0.17</td>
</tr>
<tr>
<td>XLM</td>
<td>93.66 ± 0.13</td>
</tr>
</tbody>
</table>

Table 6: Accuracy results for cross-lingual experiments from Czech to English.

Based on the results, we can conclude that the XLM-R-Large model is very capable of transferring knowledge between English and Czech (and probably between other languages as well). It is also important to note that Czech and English are languages from a different language family with a high number of differences both in syntax and grammar.

5.3 Discussion & Remarks

We can see from the results that the recent pre-trained transformer-based models beat the older approaches (lrc and LSTM) by a large margin. The monolingual Czert-B model is in general outperformed only by the XLM-R-Large and XLM-R-Base models, but these models have five times/three times more parameters, and eight times larger vocabulary. Taking into account these facts, the Czert-B model is still very competitive. It may be beneficial in certain situations to use a smaller model like this that does not need such computational resources as the ones that are required by the XLM-R-Large.

During the fine-tuning, we observed that in most cases, the lower learning rate 2e-6 (see Table 7 in Appendix A.2) leads to better results. Thus we recommend using the same one or similar order. The higher learning rates tend to provide worse results and the model does not converge.

According to the generally higher confidence interval, the fine-tuning of a smaller dataset like FB that has only about 6k training examples is, in general, less stable and more prone to overfitting than training a model on datasets with tens of thousands of examples. We also noticed that fine-tuning of the mBERT and SlavicBERT on the Mallcz dataset is very unstable (see the confidence interval in Table 3). Unfortunately, we did not find out the reason. A more detailed error analysis could reveal the reason.

6 Conclusion

In this work, we evaluated the performance of available transformer-based models for the Czech language on the task of polarity detection. We compared the performance of the monolingual and multilingual models and we showed that the large XLM-R-Large model can outperform the monolingual Czert-B model. The older approach based on recurrent neural networks is surpassed by a very large margin by the Transformers. Moreover, we achieved new state-of-the-art results on all three Czech polarity detection datasets.

We performed zero-shot cross-lingual polarity detection from English to Czech (and vice versa) with four multilingual models. We showed that the XLM-R-Large is able to detect polarity in another language without any labeled data. The model performs no worse than 4.4% in comparison to our new state-of-the-art monolingual model. To the best of our knowledge, this is the first work that aims at cross-lingual polarity detection in Czech. Our code and pre-trained models are publicly available.

In the future work, we intend to perform a deep error analysis to find in which cases the current models fail and compare approaches that use the linear cross-lingual transformations (Artetxe et al., 2018; Brychcin, 2020) that explicitly map semantic spaces into one shared space. The second step in the cross-lingual settings is to employ more than two languages and utilize the models for different domains.

Acknowledgments

This work has been partly supported by ERDF "Research and Development of Intelligent Components of Advanced Technologies for the Pilsen Metropolitan Area (InteCom)" (no.: CZ.02.1.01/0.0/0.0/17 048/0007267); and by Grant No. SGS-2019-018 Processing of heterogeneous data and its specialized applications. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-
INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

References


A Appendix

A.1 LSTM Hyper-parameters
We use cross-entropy as the loss function and the Adam (Kingma and Ba, 2014) optimizer with default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$) and with a modification from (Loshchilov and Hutter, 2017) for the FB dataset. The embedding layer is trainable with a maximum size of 300k. The max sequence length for the input $t$ tokens is 64 for the FB dataset and 512 for the CSFD and Mallcz dataset with weight decay in the optimizer set to 0. We use Czech pre-trained fastText (Bojanowski et al., 2017) embeddings\textsuperscript{17}. For the Mallcz and CSFD datasets, we use 128 units in the BiLSTM layers and a batch size of 128. For the FB dataset, we use 256 units in the BiLSTM layers and a batch size of 256 with weight decay in the optimizer set to 1e-4.

For all datasets, we use 10% of total steps (batch updates) to warm up the learning rate, which means that during the training, the linear rate is firstly linearly increasing to the initial learning rate before being decayed with the corresponding learning rate scheduler. The dropout after the BiLSTM layers is set to 0.2. We use cosine (the * symbol in Table 7) and the exponential learning rate scheduler (the ‡ symbol in Table 7) with a decay rate set to 0.05. Table 7 contains the initial learning rate and the number of epochs for the LSTM model for each dataset.

A.2 Transformer Hyper-parameters
For fine-tuning of the transformer-based models, we use the same modification (Loshchilov and Hutter, 2017) of the Adam (Kingma and Ba, 2014) optimizer with default weight decay set to 1e-2. We use different learning rates and a number of epochs for each combination of the models and datasets, see Table 7. We use either constant linear rate or linear learning rate decay without learning rate warm-up. We use default values of all other hyper-parameters.

For the cross-lingual experiments, we use only the linear learning rate decay scheduler with the initial learning rate set to 2e-6 without learning rate warm-up. For the cross-lingual experiments from English to Czech, the numbers of epochs used for the fine-tuning are 25, 5 and 9 for the XLM-R-Base, XLM-R-Large and XLM models\textsuperscript{18}, respectively.

A.3 Computational Cluster
For fine-tuning of the Transformers models we use the Czech national cluster Metacentrum\textsuperscript{19}. We use two NVIDIA A100 GPUs each with 40GB memory.

\textsuperscript{17}Available at https://fasttext.cc/docs/en/crawl-vectors.html
\textsuperscript{18}The mBERT model did not converge for this experiment
\textsuperscript{19}See https://wiki.metacentrum.cz/wiki/Usage_rules/Acknowledgement

Figure 1: Subword token histograms for the CSFD and Mallcz datasets for the Czert-B model.
Table 7: The final monolingual results as macro $F_1$ score and hyper-parameters for all three Czech polarity datasets on two and three classes. For experiments with neural networks performed by us, we present the results with a 95% confidence interval. For each result, we state the used learning rate and the number of epochs used for the training. The $‡$ symbol denotes that the result was obtained with constant learning rate, $∗$ denotes the cosine learning rate decay, $†$ denotes exponential learning rate decay, otherwise the linear learning rate decay was used.

<table>
<thead>
<tr>
<th>Model</th>
<th>CSFD</th>
<th>FB</th>
<th>Malcz</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM (ours)</td>
<td>79.88 ± 0.18 (2e-6 / 2)</td>
<td>72.89 ± 0.49 (2e-6 / 5)</td>
<td>73.43 ± 0.12 (2e-6 / 10)</td>
</tr>
<tr>
<td>XLM-R-Large</td>
<td>76.56 ± 0.12 (2e-5 / 12)</td>
<td>77.05 ± 0.15 (2e-6 / 5)</td>
<td>77.02 ± 0.37 (2e-6 / 10)</td>
</tr>
<tr>
<td>XLM-R-Base</td>
<td>76.56 ± 0.10 (2e-5 / 12)</td>
<td>76.90 ± 0.38 (2e-6 / 5)</td>
<td>79.35 ± 0.24 (2e-6 / 10)</td>
</tr>
<tr>
<td>mBERT</td>
<td>82.74 ± 0.16 (2e-6 / 13)</td>
<td>71.61 ± 0.13 (2e-6 / 15)</td>
<td>70.79 ± 0.54 (2e-6 / 10)</td>
</tr>
<tr>
<td>mBERT</td>
<td>82.59 ± 0.12 (2e-6 / 12)</td>
<td>72.93 ± 0.53 (2e-6 / 4)</td>
<td>73.34 ± 2.54 (2e-6 / 10)</td>
</tr>
<tr>
<td>RandomALBERT</td>
<td>75.79 ± 0.18 (2e-6 / 14)</td>
<td>62.53 ± 0.46 (2e-6 / 14)</td>
<td>64.81 ± 0.25 (2e-6 / 15)</td>
</tr>
<tr>
<td>XLM-R-Base</td>
<td>84.82 ± 0.10 (2e-6 / 15)</td>
<td>77.81 ± 0.50 (2e-6 / 7)</td>
<td>75.43 ± 0.07 (2e-6 / 15)</td>
</tr>
<tr>
<td>XLM-R-Large</td>
<td>87.08 ± 0.11 (2e-6 / 11)</td>
<td>81.70 ± 0.64 (2e-6 / 4)</td>
<td>79.81 ± 0.21 (2e-6 / 20)</td>
</tr>
<tr>
<td>mBERT</td>
<td>83.67 ± 0.11 (2e-6 / 11)</td>
<td>71.46 ± 1.58 (2e-6 / 9)</td>
<td>77.56 ± 0.08 (2e-6 / 15)</td>
</tr>
</tbody>
</table>

Table 8: Comparison of results from (Sido et al., 2021) with results obtained by us.

<table>
<thead>
<tr>
<th>Model</th>
<th>CSFD</th>
<th>FB</th>
<th>Malcz</th>
</tr>
</thead>
<tbody>
<tr>
<td>mBERT</td>
<td>82.80 ± 0.14 (2e-6 / 13)</td>
<td>82.74 ± 0.16 (2e-6 / 13)</td>
<td>71.72 ± 0.91 (2e-5 / 6)</td>
</tr>
<tr>
<td>SlavicBERT</td>
<td>82.51 ± 0.14 (2e-6 / 12)</td>
<td>82.59 ± 0.12 (2e-6 / 12)</td>
<td>73.87 ± 0.91 (2e-5 / 8)</td>
</tr>
<tr>
<td>RandomALBERT</td>
<td>75.40 ± 0.18 (2e-6 / 13)</td>
<td>75.79 ± 0.18 (2e-6 / 14)</td>
<td>59.50 ± 0.47 (2e-6 / 14)</td>
</tr>
<tr>
<td>Czert-A</td>
<td>79.58 ± 0.46 (2e-6 / 8)</td>
<td>79.89 ± 0.60 (2e-6 / 8)</td>
<td>72.47 ± 0.72 (2e-5 / 8)</td>
</tr>
<tr>
<td>Czert-B</td>
<td>84.79 ± 0.26 (2e-5 / 12)</td>
<td>84.80 ± 0.10 (2e-5 / 12)</td>
<td>76.55 ± 0.14 (2e-6 / 12)</td>
</tr>
</tbody>
</table>

Figure 2: Subword token histograms for the CSFD and Malcz datasets for the XLM-R-Base and XLM-R-Large models.

Figure 3: Subword token histograms for the CSFD and Malcz datasets for the mBERT model.
Metric Learning in Multilingual Sentence Similarity Measurement for Document Alignment

Charith Rajitha, Lakmali Piyarathne, Dilan Sachintha, Surangika Ranathunga
Department of Computer Science and Engineering, University of Moratuwa, Katubedda 10400, Sri Lanka.
[rajitha.16,lakmali.16,dilansachintha.16,surangika]@cse.mrt.ac.lk

Abstract

Document alignment techniques based on multilingual sentence representations have recently shown state of the art results. However, these techniques rely on unsupervised distance measurement techniques, which cannot be fine-tuned to the task at hand. In this paper, instead of these unsupervised distance measurement techniques, we employ Metric Learning to derive task-specific distance measurements. These measurements are supervised, meaning that the distance measurement metric is trained using a parallel dataset. Using a dataset belonging to English, Sinhala, and Tamil, which belong to three different language families, we show that these task-specific supervised distance learning metrics outperform their unsupervised counterparts, for document alignment.

1 Introduction

Document alignment is an important precursor to build parallel corpora from noisy data sources. Document alignment is also useful in multilingual Information Retrieval, as well as for tasks such as false news detection across different languages. Traditionally, document alignment has been achieved by metadata-based methods (Resnik, 1998, 1999) and translation based methods (Uszkoreit et al., 2010; Dara and Lin, 2016). Metadata-based methods rely on exploiting the meta information of the selected data sources, which may be inconsistent across different sources. On the other hand, translation based methods assume the availability of a Machine Translation (MT) system.

To overcome these issues, recent research has exploited the use of multilingual sentence representations (multilingual sentence embeddings) such as LASER (Artetxe and Schwenk, 2019)\(^1\). Here, vector representations are derived for documents in both source and target languages. Then, for a given document in the source side, the most similar counterpart is identified from the target side. Euclidean distance and cosine distance are used in existing document alignment systems (Uszkoreit et al., 2010; El-Kishky and Guzmán, 2020). However, these similarity metrics cannot be fine-tuned for the selected task or data at hand. The alternative is to use Metric Learning, which focuses on constructing a problem-specific distance metric automatically from data (de Vazelhes et al., 2019). Metric Learning-based distance measurement techniques have been successfully employed in image classification and image identification tasks (Pacchiardi et al., 2021). In this paper, we apply two such Metric Learning algorithms on multilingual sentence representations to identify similar document pairs in a bilingual setting. We experimented with Sparse High-Dimensional Metric Learning (SDML) (Qi et al., 2009), and Information Theoretic Metric Learning (ITML) (Davis et al., 2007) algorithms. To the best of our knowledge, this is the first work to exploit the use of Metric Learning with respect to multilingual sentence representations for the purpose of document alignment.

El-Kishky and Guzmán (2020)’s system was used as the baseline. This research can be considered as the best in this line of research at the moment. They used sentence embeddings using the LASER toolkit. They used Euclidean similarity as the sentence similarity measurement. Weighted sentence similarity scores were aggregated to derive the document similarity. In this work, we replaced Euclidean distance measurement with Metric Learning based distance measurement. We used a dataset that contains documents crawled from news websites belonging to three languages Sinhala (Si), Tamil (Ta), and English (En), which belong to three different language families (Indo-Aryan, Dravidian, and Indo-European, respectively) (Sachintha et al., 2021). Note

\(^1\)https://github.com/facebookresearch/LASER

1154

Proceedings of Recent Advances in Natural Language Processing, pages 1154–1161
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_130
that Sinhala is a low-resource language. In the recent language categorization by Joshi et al. (2020), Sinhala belongs to class 0, meaning that it has exceptionally limited resources. Tamil is considered as a medium-resourced language. It is categorised as 3, meaning that it has a strong web presence, and a cultural community that backs it. We separately report results for the three language pairs, En-Si, En-Ta and Si-Ta. For all the document alignment tasks except one news dataset in En-Si, Metric Learning based distance measurement performed better than the unsupervised distance measurement techniques. To train the Metric Learning models, a very small parallel corpus of 5000 words were used. Further experiments showed that the content (i.e. the domain) in the parallel corpus has minimal impact on the performance of Metric Learning models. The above results were obtained using LASER embeddings, to be a fair comparison with El-Kishky and Guzmán (2020). We experimented with XLM-R multilingual embeddings (Conneau et al., 2020) as well, using the ITML Metric Learning model. Since XML-R showed superior performance over LASER, we conducted extensive experiments where the ITML model was built using different amounts of parallel data with respect to XLM-R. Our results show that the ITML model trained with XLM-R embeddings works equally well even with 1000 parallel sentences. Our code is publicly released.

## 2 Related Work

### 2.1 Early Document Alignment Systems

Early work on document alignment mainly relied on metadata based (Resnik, 1998, 1999) and translation based approaches (Dara and Lin, 2016). Even though most metadata based systems are language independent, these methods tend to have lower results due to the inconsistency of these metadata across different data sources. Translation based approaches outperformed metadata based systems since they depend on the textual context of the documents (Vos, 2004). However the accuracy of these highly depends on the used MT system.

### 2.2 Multilingual Embedding Based Document Alignment Systems

Guo et al. (2019) extended the Hierarchical Attention Networks (HAN) architecture (Yang et al., 2016) for parallel document mining. They compared performance of the HAN architecture for document alignment with a neural Bag of Words (BoW) document embedding model, where document embeddings were generated by simply averaging multilingual sentence embeddings. El-Kishky and Guzmán (2020) proposed a method that uses the LASER toolkit to create the multilingual sentence embeddings. They calculated the distance between each pair of sentence representations of the source and target documents, and took the sum of the calculated distances for sentence pairs with a weighting value to calculate the distance between two documents. Euclidean distance was used as the distance metric for calculating the distance between sentence embeddings. Greedy movers distance algorithm (El-Kishky and Guzmán, 2020), which is an improved version of the Earth Movers Distance algorithm (Rubner et al., 1998), was used to take the sum of distance between sentences. They have used multiple sentence weighting schemes such as sentence length, inverse document frequency (IDF) and sentence length combined with IDF (SLIDF) to improve the results further. However, their dataset is not publicly available.

### 2.3 Multilingual Contextual Embedding Models

The LASER model consists of a single encoder implemented using a biLSTM (bi-Long Short Term Memory) network, which can handle multiple languages. This guarantees that sentences that are semantically similar lie closer in the embedding space. This encoder is coupled with an auxiliary decoder, and is pre-trained on 93 languages (using parallel corpora). Sentence embeddings are obtained by applying a max-pooling operation over the output of the encoder and used to initialize the decoder LSTM through a linear transformation. The encoder and decoder are shared by all the languages and for that, a joint byte-pair encoding (BPE) vocabulary made on the concatenation of all training corpora was used. The XLM-R model is based on a Transformer model (Vaswani et al., 2017). It has an encoder

---

1. Related Work
2. Early Document Alignment Systems
3. Multilingual Embedding Based Document Alignment Systems
4. Multilingual Contextual Embedding Models
5. Multilingual Contextual Embedding Models
trained only with a masked language model objective. In essence, XLM-R is the multilingual version of RoBERTa (Liu et al., 2019). RoBERTa improves on the popular BERT model (Devlin et al., 2019) with more data, larger batch sizes, longer training times, training on longer sequences and dynamically changing the masking pattern applied to the training data. XLM-R is trained with common crawl data from 100 languages, and has shown to outperform mBERT on multiple Natural Language Processing tasks.

3 Metric Learning

Unsupervised metrics such as Euclidean or cosine are commonly used to calculate the distance between two embeddings or vectors. This is the same for sentence embeddings. However, these unsupervised metrics cannot be optimised for the particular data set or the task. In contrast, Metric Learning uses a supervised algorithm to learn the best distance measurement metrics that are task specific, using the given data. Metric Learning has been successfully used for tasks such as pattern recognition and face identification in the field of image classification (Pacchiardi et al., 2021; Wright et al., 2010; Xiang et al., 2008). All the Metric Learning algorithms use Mahalanobis distance as the distance metric. Mahalanobis distance between two points \( x, x' \) is defined as in Equation 1,

\[
D_L(x, x') = \sqrt{(Lx - Lx')^T(Lx - Lx')} 
\]  

This can also be written as shown in Equation 2,

\[
D_L(x, x') = \sqrt{(x - x')^TMM(x - x')} 
\]

where \( M = L^TL \). The Metric Learning algorithms calculate the matrix L according to the given training data set.

Commonly used Metric Learning algorithms include: Neighborhood Components Analysis (NCA) (Goldberger et al., 2005), Large Margin Nearest Neighbors (LMNN) (Weinberger et al., 2005), Sparse High-Dimensional Metric Learning (SDML) (Qi et al., 2009), and Information Theoretic Metric Learning (ITML) (Davis et al., 2007). ITML algorithm is able to learn a distance function that generalizes well to unseen test data and is also fast and scalable. Davis et al. (2007) used LogDet divergence for regularization of Mahalanobis distance learning methods. This regularization aims at keeping the learned distance close to the Euclidean distance. It has been shown that ITML converges to a global optimal solution. SDML algorithm aims on training an accurate distance metric for high dimensional data from a small sample. In addition to the LogDet Divergence regularization introduced by Davis et al. (2007), L1-regularization on the off diagonal elements of the matrix was used to reduce the chance of over fitting.

4 Methodology

4.1 Baseline Document Alignment System

The document alignment system proposed by El-Kishky and Guzmán (2020) was used as the baseline, which was discussed in Section 2. In this method, first the sentence representations are derived from a multilingual contextual embedding model. The algorithm first calculates the Euclidean distance between each sentence pair from source to target. The calculated weight of the sentence is used as the weights (aka mass) when calculating the Greedy Movers Distance between document pairs. Then the minimum value of the weights is selected as the flow value. Then the distance of the sentence pair is updated by adding the multiplication of flow value and the subtraction of two weight values (\( |s_A - s_B| \)) as shown in Equation 3.

\[
distance = distance + ||s_A - s_B|| \times flow 
\]

In this method, each document is represented as a normalized bag of sentences. Instead of assigning equal weights to each sentence, they have used three weighting schemes for sentences: (1) sentence length (SL) weighting, (2) Inverse Document Frequency (IDF) weighting and (3) sentence length and IDF (SLIDF) weighting.

4.1.1 Sentence Length Weighting

It is common to have a great variance in the length of the sentences in a given document. Longer sentences tend to contain more content than the shorter ones. Therefore, longer sentences should have more weight when aligning documents. Sentence Length (SL) weighting scheme uses the sentence length for consideration when calculating the weights.

In the SL weighting scheme, each sentence is weighted using the ratio between the number of tokens into the length of the sentence and the sum of number of tokens into the length of the sentence for each sentence. For the \( i^{th} \) sentence in document...
A, weight $d_{A,i}$ is calculated using the equation 4.

$$d_{A,i} = \frac{\text{count}(i) \times |i|}{\sum_{s \subseteq A} \text{count}(s) \times |s|} \quad (4)$$

where $|i|$, $|s|$ - Number of tokens in sentence $i$ and $s$ respectively.

### 4.1.2 IDF Weighting

The IDF weighting scheme is a common weighting scheme in Information Retrieval. If some content is occurring a number of times in every document, that content could be less semantically informative. This could be the titles, and other website specific text. Using this fact, these sentences should get a less weight than other sentences. IDF Weighting scheme is calculated using the equation 5,

$$d_{A,i} = 1 + \log \frac{N + 1}{1 + |d \subseteq D : s \subseteq d|} \quad (5)$$

where $N$ - Total number of documents and $|d \subseteq D : s \subseteq d|$ - Number of documents in which sentence $s$ occurs.

### 4.2 Supervised Distance Metric

As our main contribution, we used supervised distance metrics calculated using Metric Learning algorithms instead of using unsupervised metric used by El-Kishky and Guzmán (2020).

We used both Sparse High-Dimensional Metric Learning (SDML) (Qi et al., 2009) and Information Theoretic Metric Learning (ITML) algorithms (Davis et al., 2007; de Vazelhes et al., 2019) for our experiments. To provide a supervised signal to the Metric Learning algorithms, a parallel corpus was used. The sentences in the parallel corpus were converted to embedding pairs. The Metric Learning models were trained using these embeddings. Once the model is trained, it is able to provide the distance value between new sentence embedding pairs. We replaced the unsupervised sentence distance calculation method used in the baseline with these trained supervised distance metrics to calculate the distance between documents.

### 4.3 Date-wise filtering

In most of the cases, an article corresponding to a news is published in all languages within the same day. Therefore, using this property of news websites, we reduced the search space to a date from the whole web domain. Then the news items from a particular day were selected and aligned.

5 Data Set

We used the dataset presented by Sachintha et al. (2021). This dataset contains news articles belonging to four news websites published in English, Tamil and Sinhala languages. From each of the websites, around 2000 documents (per language) were selected based on the published date of the articles. Table 1 shows the statistics of the dataset used for document alignment. They have identified different characteristics in these articles and have used those to filter out the ground truth alignment. In addition, they have manually aligned the dataset with the help of human annotators. The aligned dataset was verified by the same annotators by switching the data sets.

5000 sentence pairs from the parallel corpora published by Fernando et al. (2020) were used to train the Metric Learning models. This parallel dataset is specific to the parallel data has any impact on Metric Learning, a 5000 subset from the Open Subtitle Parallel Corpus (Lison and Tiedemann, 2016) was used for En-Si. This corpus consists of translations of movie subtitles.

6 Experiments

For our experiments we used the python implementation of the two metric learning algorithms (de Vazelhes et al., 2019).

The first experiment was to determine whether Metric Learning based distance measurement could outperform the unsupervised counterpart. To be a fair comparison, we used the LASER embedding model as used by El-Kishky and Guzmán (2020). Euclidean and Cosine unsupervised distance metrics, as well as ITML and SDML Metric Learning based distance metrics were calculated for both SL and IDF weighting schemes. 5000 parallel sentences from Fernando et al. (2020)’s parallel corpora were used to train the Metric Learning models for En-Ta and Si-Ta. For En-Si, SDML and ITML were trained with SL weighting for Fernando et al. (2020)’s parallel corpus and the open subtitle corpus. IDF was trained with the latter corpus only. This is to measure the impact of the type of parallel corpus on the Metric Learning models.

We conducted another experiment to compare the performance of LASER and XLM-R multilingual embedding models. First, the same 5000 sentence pairs from Fernando et al. (2020) were used to ob-
<table>
<thead>
<tr>
<th>Web site</th>
<th>Sinhala - English</th>
<th>Tamil - English</th>
<th>Sinhala - Tamil</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sinhala  English</td>
<td>GA</td>
<td>Tamil  English</td>
</tr>
<tr>
<td>Army News</td>
<td>2033  2081</td>
<td>1848</td>
<td>1905  2081</td>
</tr>
<tr>
<td>Hiru</td>
<td>3133  1634</td>
<td>1397</td>
<td>2886  1634</td>
</tr>
<tr>
<td>ITN</td>
<td>4898  1942</td>
<td>352</td>
<td>1521  1942</td>
</tr>
<tr>
<td>News First</td>
<td>1819  2278</td>
<td>344</td>
<td>2333  2278</td>
</tr>
</tbody>
</table>

Table 1: Document alignment data set with golden alignment counts. GA - Golden Alignment

<table>
<thead>
<tr>
<th>Language</th>
<th>Weighting Scheme</th>
<th>Distance Metric</th>
<th>News Site</th>
</tr>
</thead>
<tbody>
<tr>
<td>En - Si</td>
<td>SL</td>
<td>Euc</td>
<td>Hiru  ITN</td>
</tr>
<tr>
<td>IDF</td>
<td></td>
<td>0.82  0.85  0.88 0.99</td>
<td>0.82  0.82 0.91 0.99</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.82  0.82 0.87 0.90</td>
<td>0.84  0.87 0.89 0.99</td>
</tr>
<tr>
<td></td>
<td>SDML-OPUS-Subtitle Corpus</td>
<td>0.85 0.87 0.90</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>SDML - NLPC Corpus</td>
<td>0.84 0.87</td>
<td>0.89 0.99</td>
</tr>
<tr>
<td></td>
<td>ITML-OPUS-Subtitle Corpus</td>
<td>0.84 0.85 0.89</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>ITML - NLPC Corpus</td>
<td>0.83 0.85 0.88</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>Euc</td>
<td>0.78 0.84 0.81</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>Cosine</td>
<td>0.77 0.80 0.82</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>SDML-OPUS-Subtitle Corpus</td>
<td>0.85 0.89 0.87</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>ITML-OPUS-Subtitle Corpus</td>
<td>0.82 0.84 0.86</td>
<td>0.98</td>
</tr>
<tr>
<td>En - Ta</td>
<td>SL</td>
<td>Euc</td>
<td>Hiru  ITN</td>
</tr>
<tr>
<td>IDF</td>
<td></td>
<td>0.26 0.44 0.41 0.69</td>
<td>0.30 0.47 0.50 0.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.30 0.47 0.50 0.78</td>
<td>0.41 0.66 0.62 0.89</td>
</tr>
<tr>
<td></td>
<td>SDML - NLPC Corpus</td>
<td>0.41 0.66 0.62 0.89</td>
<td>0.48 0.67 0.67 0.91</td>
</tr>
<tr>
<td></td>
<td>ITML - NLPC Corpus</td>
<td>0.48 0.67 0.67 0.91</td>
<td>0.24 0.50 0.37 0.57</td>
</tr>
<tr>
<td></td>
<td>Euc</td>
<td>0.24 0.50 0.37 0.57</td>
<td>0.27 0.52 0.45 0.68</td>
</tr>
<tr>
<td></td>
<td>Cosine</td>
<td>0.27 0.52 0.45 0.68</td>
<td>0.46 0.64 0.56 0.82</td>
</tr>
<tr>
<td></td>
<td>SDML - NLPC Corpus</td>
<td>0.46 0.64 0.56 0.82</td>
<td>0.43 0.66 0.59 0.84</td>
</tr>
<tr>
<td></td>
<td>ITML - NLPC Corpus</td>
<td>0.43 0.66 0.59 0.84</td>
<td>0.45 0.64 0.60 0.88</td>
</tr>
<tr>
<td>Si - Ta</td>
<td>SL</td>
<td>Euc</td>
<td>Hiru  ITN</td>
</tr>
<tr>
<td>IDF</td>
<td></td>
<td>0.45 0.41 0.63 0.83</td>
<td>0.50 0.64 0.60 0.88</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.45 0.41 0.63 0.83</td>
<td>0.50 0.64 0.60 0.88</td>
</tr>
<tr>
<td></td>
<td>SDML - NLPC Corpus</td>
<td>0.57 0.61 0.74 0.91</td>
<td>0.51 0.47 0.64 0.88</td>
</tr>
<tr>
<td></td>
<td>ITML - NLPC Corpus</td>
<td>0.51 0.47 0.64 0.88</td>
<td>0.42 0.47 0.59 0.73</td>
</tr>
<tr>
<td></td>
<td>Euc</td>
<td>0.42 0.47 0.59 0.73</td>
<td>0.44 0.61 0.59 0.77</td>
</tr>
<tr>
<td></td>
<td>Cosine</td>
<td>0.44 0.61 0.59 0.77</td>
<td>0.53 0.64 0.71 0.86</td>
</tr>
<tr>
<td></td>
<td>SDML - NLPC Corpus</td>
<td>0.53 0.64 0.71 0.86</td>
<td>0.47 0.61 0.60 0.80</td>
</tr>
<tr>
<td></td>
<td>ITML - NLPC Corpus</td>
<td>0.47 0.61 0.60 0.80</td>
<td>0.47 0.61 0.60 0.80</td>
</tr>
</tbody>
</table>

Table 2: Recall values for Document Alignment with LASER embeddings

In order to determine the impact of the parallel dataset size on the performance of Metric Learning models, we trained separate ITML models using 1000, 2000, 3000 and 5000 parallel sentences for all three languages using Fernando et al. (2020)'s corpus. Embeddings were generated with XLM-R.

7 Results

We followed the method used for document alignment evaluation in WMT16 document alignment shared task (Buck and Koehn, 2016). The ground truth documents only contain a small fraction of parallel documents. There can be many more valid cross lingual document pairs in the dataset. Therefore we evaluated the aligned document pairs using recall (i.e. what percentage of the aligned document pairs in the golden alignment set are found by the algorithm) from the ground truth.
Table 3: Recall of Document Alignment for XLM-R Embeddings with ITML for different parallel dataset sizes

<table>
<thead>
<tr>
<th>Language</th>
<th>Parallel Corpus Size</th>
<th>News Site</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hiru</td>
<td>ITN</td>
</tr>
<tr>
<td>En - Si</td>
<td>1000</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>3000</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>0.91</td>
</tr>
<tr>
<td>En - Ta</td>
<td>1000</td>
<td>0.82</td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td>0.83</td>
</tr>
<tr>
<td></td>
<td>3000</td>
<td>0.83</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>0.82</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>0.82</td>
</tr>
<tr>
<td>Si - Ta</td>
<td>1000</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>3000</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>0.79</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>0.79</td>
</tr>
</tbody>
</table>

data-set, as done by Dara and Lin (2016). Results of our system against El-Kishky and Guzmán (2020)’s system on LASER embeddings are given in Table 2. It also shows the results of the two parallel datasets used to build the Metric Learning models for En-Si. How the performance of our system varies with respect to the size of the parallel dataset used with the XLM-R model is given in Table 3. The results related to all the distance measurement techniques depend on the news source. If most of the news items are present in all the three languages, this sends a strong signal to the aligner. A very good example is the En-Si pair in Army News. Number of Sinhala and English documents in this source is roughly equal. The results confirm that Metric Learning based distance measurement has been very effective in aligning documents for all three language pairs. Even for the Newsfirst data source for the En-Si pair, metric learning (SDML) lags behind cosine similarity only by a very small margin. It is evident that XLM-R significantly outperforms LASER. XLM-R performing very well even for 1000 parallel sentence is very promising - this means that Metric Learning could be employed with languages included in the XLM-R model as long they have a very small parallel corpus. Another factor we wanted to investigate is the impact of the level of language representation in the multilingual embedding model on document alignment. Although LASER and XLM-R include data from multiple languages, they do not include data from all the languages in equal amounts. If we hypothesize that the amount of language data is proportional to the language categorization proposed by Joshi et al. (2020), results related to Sinhala has to be the lowest. However, the results do not convey this message. Thus we believe that factors such as language relatedness and language complexity might be playing a role here.

8 Conclusion

This paper presented the use of supervised distance measurement techniques on multilingual sentence embedding based document similarity calculation. Results show that this supervised approach is being able to consistently outperform the unsupervised counterparts for three document alignment tasks. Only a small parallel corpus is required to train the Metric Learning distance measurements. Thus, these techniques can be employed with respect to low-resource languages as well. The main drawback of Metric Learning algorithms is that they are very much computationally expensive. Therefore, in the future, we plan to optimize these algorithms to achieve better efficiency.

Acknowledgments

This publication was funded by a Senate Research Committee (SRC) Grant of University of Moratuwa, Sri Lanka.
References


Multi-label Diagnosis Classification of Swedish Discharge Summaries –
ICD-10 Code Assignment Using KB-BERT

Sonja Remmer\textsuperscript{a,b}  Anastasios Lamproudis\textsuperscript{a}  Hercules Dalianis\textsuperscript{a,b}
remmer@dsv.su.se  anastasios@dsv.su.se  hercules@dsv.su.se
\textsuperscript{a}Department of Computer and Systems Sciences (DSV)
Stockholm University, Kista, Sweden
\textsuperscript{b}Norwegian Centre for E-health Research, Tromsø, Norway

Abstract

The International Classification of Diseases (ICD) is a system for systematically recording patients’ diagnoses. Clinicians or professional coders assign ICD codes to patients’ medical records to facilitate funding, research, and administration. In most health facilities, clinical coding is a manual, time-demanding task that is prone to errors. A tool that automatically assigns ICD codes to free-text clinical notes could save time and reduce erroneous coding. While many previous studies have focused on ICD coding, research on Swedish patient records is scarce. This study explored different approaches to pairing Swedish clinical notes with ICD codes. KB-BERT, a BERT model pre-trained on Swedish text, was compared to the traditional supervised learning models Support Vector Machines, Decision Trees, and K-nearest Neighbours used as the baseline.

When considering ICD codes grouped into ten blocks, the KB-BERT was superior to the baseline models, obtaining an $F_1$-micro of 0.80 and an $F_1$-macro of 0.58. When considering the 263 full ICD codes, the KB-BERT was outperformed by all baseline models at an $F_1$-micro and $F_1$-macro of zero. Wilcoxon signed-rank tests showed that the performance differences between the KB-BERT and the baseline models were statistically significant.

1 Introduction

There are both administrative and statistical purposes of ICD coding. Administrative to reimburse the clinical unit or hospital, but also to plan healthcare. The codes are assigned by both treating physicians and designated coders. The current version of the ICD system, ICD-10, contains tens of thousands of codes divided into 22 chapters (WHO, 2016).

ICD coding is time-consuming and error-prone, either missing the main diagnosis or displaying errors in the coding in up to 20 per cent of the patient records (Jacobsson and Serdén, 2013). Therefore, it would be valuable to have a supporting tool to assist the physician or coder in choosing among the codes.

In this article, Swedish patient records in the medical speciality of gastrointestinal surgery and their already assigned ICD-10 codes are used to perform supervised learning to predict ICD-10 codes. More specifically, the part of the patient records that summarises the patient’s care period at the time of the discharge, the discharge summaries, and their associated ICD-10 codes are used. The assigned codes belong to the Swedish version of the ICD-10 system known as ICD-10-SE (Socialstyrelsen, 2018). The codes considered are both full ICD codes at the highest level of granularity and the full codes grouped into ten blocks. The research question is how the deep learning language model KB-BERT, compared to the traditional supervised learning models Support Vector Machines, Decision Trees, and K-Nearest Neighbours performs in pairing discharge summaries with the correct ICD codes.

2 Related Research

ICD coding has been a popular research area for decades. The interest increased with a public challenge hosted by the Computational Medical Center called the 2007 Computational Medicine Challenge, where contestants were asked to create a system for pairing radiology reports with the correct ICD codes. Most submitted solutions used hand-crafted rules, traditional supervised learning models such as Support Vector Machines, or a combination of these two approaches (Pestian et al., 2007). One of the top-performing systems used a combination of rule-based and machine learning elements, achieving an $F_1$-micro score of 0.89 by utilising Decision Trees to generate rules automati-
Since 2007, ICD classification has shifted away from rule-based techniques, and many recent studies use traditional supervised learning methods or deep learning approaches. Examples of conventional models used in previous ICD coding papers are Support Vector Machines, Decision Trees, K-nearest Neighbours, Naïve Bayes, and ensembles of these models. In Kaur and Ginige (2018), comparing these conventional models with Multi-layer Perceptrons resulted in Decision Trees and AdaBoost using Decision Trees being the superior classifiers at F-scores of approximately 0.9. Hasan et al. (2016) compared traditional models with Convolutional Neural Networks and concluded that the results for the Convolutional Neural Networks were comparable with the results of Support Vector Machines, but that Support Vector Machines outperformed Convolutional Neural Networks as the number of classes increased. The best-achieved accuracy score of Support Vector Machines in (Hasan et al., 2016) was 0.75. For Bulgarian, Boytcheva (2011) carried out ICD classification using Support Vector Machines. She used 6,200 and 1,300 electronic patient records for training and evaluation, respectively, obtaining a precision of 0.97, a recall of 0.74, and an F-score of 0.85.

An increasingly popular deep learning approach to ICD coding tasks is using the language model BERT (Bidirectional Encoder Representations from Transformers) developed in 2018 by Devlin et al. (2019). BERT was pre-trained on 3.3 billion words from two English corpora – the BooksCorpus and the English Wikipedia – making it an expert in understanding general English. However, BERT has also been adapted to domain-specific language. For example, Lee et al. (2020) developed BioBERT – a BERT model pre-trained on biomedical texts. When Amin et al. (2019) adopted BioBERT to perform ICD coding, they reached an F1-micro score of 0.73. Moreover, a BERT model pre-trained on clinical text was developed by Alsentzer et al. (2019) and used by Biseda et al. (2020) for ICD classification, achieving an F1-score of 0.75.

Since the original BERT and many of its domain-specific adaptations are trained on English texts, BERT has also been adapted to understand other languages. In 2020, the National Library of Sweden pre-trained a BERT model on billions of Swedish words, naming this model KB-BERT, (Malmsten et al., 2020). Malmsten et al. (2020) showed that KB-BERT outperformed the multi-lingual version of the BERT model.

As discussed in a review paper by Stanfill et al. (2010), it is difficult to compare previous ICD classification approaches since previous studies using the techniques apply them differently. Previous studies use different label sets, evaluate the classifiers’ performance differently, and use texts written in different languages. Therefore, it is favourable to investigate the alternative ICD coding methods in the context they will be used. Moreover, while many studies explore rule-based methods, traditional supervised models, and deep learning approaches to solve ICD coding tasks, few studies use Swedish data.

Henriksson et al. (2011) attempted automatic ICD classification on Swedish data using co-occurrences of words and ICD codes. Pairing clinical notes with semantically correlated ICD codes resulted in the correct ICD code being present in the top ten suggested ICD codes in 20 per cent of the cases. When considering codes at a lower level of granularity, the correct ICD codes were found in the top 10 suggested codes in 77 per cent of the cases. Optimising the dimensionality improved these results by 18 percentage points (Henriksson and Hassel, 2013).

This study explores conventional supervised learning methods and the deep learning Swedish model KB-BERT for Swedish ICD classification. The F1-micro is used to evaluate the different approaches. The F1-macro is also presented. A discussion of the results follows, addressing the implications of the study.

3 Methodology

3.1 Data

3.1.1 ICD Codes

The ICD system is hierarchical, and at the highest level of granularity, the letter initiating each code is followed by three digits. In Figure 1, the anatomy of ICD codes is displayed.

![Figure 1: The anatomy of ICD codes.](image-url)
Health personnel assign three-digit codes like the one exemplified in Figure 1 to the patient records, and a clinical coding tool would benefit from suggesting these full codes. However, since many codes only have a few associated patient records in the data available for this study, it might not be successful to train models to predict full codes. Therefore, to give the models a fair chance to perform and, thereby, be compared, both full codes at the highest granularity level and codes grouped at a higher level are considered in this article.

The grouped codes considered are ICD codes at a two-digit level, which are known as ICD blocks. Using full ICD codes can illustrate the implications of training models when having many classes, where many of the classes have few instances. On the other hand, using ICD codes at the block level can demonstrate the implications of solving an ICD classification task at a lower level of granularity with fewer classes and more instances per class.

This paper is delimited to ICD codes related to gastrointestinal diseases. The digestive diseases reside in ICD Chapter XI, containing codes starting with a K. Chapter XI consists of ten blocks. In Table 1, the two-digit ICD codes included in each block in ICD Chapter XI and descriptions of the diseases that the blocks cover are presented.

### Table 1: ICD blocks of Chapter XI.

<table>
<thead>
<tr>
<th>ICD Block</th>
<th>Description of diseases</th>
</tr>
</thead>
<tbody>
<tr>
<td>K00-K14</td>
<td>Diseases of the oral cavity, salivary glands, jaws</td>
</tr>
<tr>
<td>K20-K31</td>
<td>Diseases of oesophagus, stomach, duodenum</td>
</tr>
<tr>
<td>K35-K38</td>
<td>Diseases of appendix</td>
</tr>
<tr>
<td>K40-K46</td>
<td>Hernia</td>
</tr>
<tr>
<td>K50-K52</td>
<td>Noninfective enteritis, colitis</td>
</tr>
<tr>
<td>K55-K64</td>
<td>Other diseases of intestines</td>
</tr>
<tr>
<td>K65-K67</td>
<td>Diseases of peritoneum</td>
</tr>
<tr>
<td>K70-K77</td>
<td>Diseases of the liver</td>
</tr>
<tr>
<td>K80-K87</td>
<td>Disorders of gallbladder, biliary tract, pancreas</td>
</tr>
<tr>
<td>K90-K93</td>
<td>Other diseases of the digestive system</td>
</tr>
</tbody>
</table>

3.1.2 Multi-label Text Classification

A hospitalised patient seldom suffers from only one disease. On the contrary, one patient can have many diagnoses, implying that one discharge summary often is paired with multiple ICD codes. Pairing one text with numerous labels is a multi-label classification task, which is different from multi-class tasks where the labels are mutually exclusive. In Figure 2, an exemplary clinical note with multiple assigned ICD codes is presented.

### Discharge summary

Tidigare helt frisk kvinna med obehag i epigastrium och rilltagna smärtor i arcus under 4 dagar. Konstaterat diaphragmafräck. Beh för misstänkt gastroenterit utan framgång. CT visade tecken på akut kolecytis och operation genomfördes med framgång. Pat hemskickad med råd att vila i minst 2 v. Fetsnål kost och mindre portioner rekommenderas.

**English translation:** Previously completely healthy woman feeling discomfort in epigastrium with increasing pain in arcus for 4 days. Confirmed diaphragmatic hernia. Unsuccessfully treated for suspected gastroenteritis. CT showed signs of acute cholecystitis. Successful operation. Pat sent home to rest for 2 w min. Low fat diet and smaller portions recommended.

**Assigned ICD codes**

- **K44.9** Diaphragmatic hernia, no obstruction or gangrene
- **K80.4** Acute cholecystitis

Figure 2: A partly made up and completely pseudonymised exemplary discharge summary.

### 3.1.3 The ICD-10 Corpus

The data used in this study is called the Stockholm EPR Gastro ICD-10 Corpus version 2 (ICD-10 Corpus)\(^1\). The ICD-10 Corpus resides in the research infrastructure Health Bank – the Swedish Electronic Health Record Bank\(^2\) which is located at the Department of Computer and Systems Sciences (DSV) at Stockholm University. Health Bank contains over 2 million electronic patient records from over 500 clinical units at Karolinska University Hospital in Stockholm between 2007 and 2014. The ICD-10 Corpus was extracted from Health Bank and consists of discharge summaries from four gastrointestinal care units.

In the ICD-10 Corpus, there are only ICD codes representing digestive diseases, which is ICD Chapter XI, containing codes starting with a K (see Table 1). Moreover, the discharge summaries were filtered to those containing more than three tokens, and discharge summaries belonging to the same patient and care period assigned the same ICD codes were merged into one discharge summary.

In Table 2, the number of discharge summaries, patients, tokens, unique tokens, full ICD codes, and ICD blocks of the ICD-10 Corpus are presented. Descriptive statistics of the number of tokens per discharge summary are available in Table 3.

---

\(^1\) This research has been approved by the Regional Ethical Review Board in Stockholm under permission no. 2007/1625-31/5.

\(^2\) See the Health Bank website ([https://dsv.su.se/healthbank](https://dsv.su.se/healthbank)) and Dalianis et al. (2015) for more information.
Table 2: Basic characteristics of the ICD-10 Corpus.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of discharge summaries</td>
<td>6,062</td>
</tr>
<tr>
<td>Number of unique patients</td>
<td>4,985</td>
</tr>
<tr>
<td>Total number of tokens</td>
<td>986,436</td>
</tr>
<tr>
<td>Number of unique tokens (vocabulary)</td>
<td>48,232</td>
</tr>
<tr>
<td>Number of unique full ICD codes</td>
<td>263</td>
</tr>
<tr>
<td>Number of unique ICD blocks</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 3: Number of tokens per discharge summary in the ICD-10 Corpus.

<table>
<thead>
<tr>
<th>Min</th>
<th>Median</th>
<th>Mean</th>
<th>Max</th>
<th>Std</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>134</td>
<td>162.7</td>
<td>1794</td>
<td>120.5</td>
</tr>
</tbody>
</table>

Table 4: Number of ICDs per discharge summary in the Full Codes and the Blocks data sets.

<table>
<thead>
<tr>
<th></th>
<th>Full codes</th>
<th>Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Median</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Mean</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>Max</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Std</td>
<td>0.5</td>
<td>0.4</td>
</tr>
</tbody>
</table>

3.2 Models

3.2.1 Baseline

A common approach to solving ICD classification tasks is using traditional supervised learning models. The traditional supervised learning models used in this study were Support Vector Machines, Decision Trees, and K-nearest Neighbours. These models were chosen since they are well-established and frequently used in related studies.

The implementations of Decision Trees and K-Nearest Neighbours used were the DecisionTreeClassifier class from the Scikit-learn library (Pedregosa et al., 2011) and the MLkNN class from the Scikit-multilearn library (Szymański and Kajdanowicz, 2018), respectively. Since the Scikit-learn implementation of Support Vector Machines (class SVC) is not directly suitable for multi-label data, one classifier per label was trained using the Scikit-learn classifier implementation of one-vs-rest (class OneVsRestClassifier). The default hyper-parameters were used.

For the baseline models to handle text input, the text has to be represented as numerical features. For this purpose, tf-idf weights as they are implemented in the Scikit-learn class TfidfVectorizer were used. tf-idf is short for term frequency-inverse document frequency and represents how important a word is in a specific document, compared to the importance of that word in all documents. Basic pre-processing steps in the form of removal of punctuation and stop words and de-capitalisation were also conducted. The list of Swedish stop words was taken from the Natural Language Toolkit (NLTK) (Bird et al., 2009).

3.2.2 KB-BERT

KB-BERT was used closely following the instructions in Devlin et al. (2019) for downstream tasks and fine-tuning. More specifically, the architecture takes advantage of the presence of a special token, namely the [CLS] (classification) token representation, used initially for the NSP (Next Sentence Prediction) task. This representation is utilised as sentence representation and is assumed to contain information describing each instance. The architecture of the KB-BERT classifier includes at its core the KB-BERT model (bert-base-swedish-cased) from which the [CLS] representation is used for each sample through a ReLU (Rectified Linear Unit) transformation as input to a fully connected classification layer.

In the spirit of Devlin et al. (2019), a minimal learning rate in the magnitude of $2 \cdot 10^{-5}$ was used. The number of warm-up steps was set to 155 for the model to approximately see all the data before the learning rate starts decaying. Due to memory constraints, the batch size of 32 was achieved using a batch size of 2 and a gradient accumulation of 16. The activation threshold binarising the floating numbers the KB-BERT outputs was set to the standard value of 0.5. Adam was used as the optimiser. The implementation of the KB-BERT classifier was done using the Transformers (Wolf et al., 2020), and Pytorch (Paszke et al., 2019) libraries.

---

3https://huggingface.co/KB/bert-base-swedish-cased
3.3 Experiment Design
To test how well the KB-BERT, Support Vector Machines, Decision Trees, and K-Nearest Neighbours perform in pairing Swedish discharge summaries with the correct ICD codes, 90 per cent of the data was used for training the models. 10-fold cross-validation was utilised to get more reliable estimates of the models’ performance and to be able to test if the observed differences in classifier performance are statistically significant. The final performance of the KB-BERT and the best performing baseline model was estimated by training on all training data and testing on the 10 per cent of the data (the held-out set) not used for comparing the classifiers.

Performance was represented by the F1-micro score. Micro averaging was chosen over macro averaging since it was considered of greater interest to train a classifier that correctly can classify as many discharge summaries as possible, rather than as many ICD codes as possible. However, macro averaged scores are presented as well.

The Wilcoxon signed-rank test (Wilcoxon, 1945) suitable for small dependent samples was used to test the statistical significance of classifier performance. The null hypotheses that the distribution of the F1-micro scores are equal were tested against the alternative hypotheses that the distribution of the F1-micro scores are not equal for the compared classifiers. The significance level was set to 0.01.

4 Results
4.1 Full ICD codes
The combined macro and micro averaged Precision (P), Recall (R), and F1-score (F1) of the KB-BERT and the baseline models during the 10-fold cross-validation when training the models using the Full codes version of the ICD-10 Corpus (263 ICD-10 codes) are presented in Table 5.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Macro</th>
<th>Micro</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>KB-BERT</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SVM</td>
<td>0.06</td>
<td>0.01</td>
</tr>
<tr>
<td>DT</td>
<td>0.10</td>
<td>0.09</td>
</tr>
<tr>
<td>KNN</td>
<td>0.11</td>
<td>0.03</td>
</tr>
</tbody>
</table>

Table 5: Combined scores for the Full codes data set during the 10-fold cross-validation.

Overall, the results were poor. All models underperformed, and the best performing model for the full codes was the Decision Trees, achieving an F1-micro of 0.29 and an F1-macro of 0.09. The KB-BERT failed to perform at all, obtaining F1-scores of zero.

As becomes evident from Figure 3, the classifier ranks remained constant during the 10-fold cross-validation, resulting in the smallest possible Wilcoxon test statistic, 0, for all pairwise comparisons of classifiers. This means that there is reason to trust that the differences in classifier performance observed in Table 5 were not only due to chance but reflect actual characteristics of the data set.

As becomes evident from Figure 3, the classifier ranks remained constant during the 10-fold cross-validation, resulting in the smallest possible Wilcoxon test statistic, 0, for all pairwise comparisons of classifiers. This means that there is reason to trust that the differences in classifier performance observed in Table 5 were not only due to chance but reflect actual characteristics of the data set.

For the KB-BERT, early stopping was used during the 10-fold cross-validation, and convergence was achieved with respect to the Binary Cross-Entropy (BCE) loss function at ten epochs during most runs. Therefore, when training the KB-BERT on all of the training data and testing it on the held-out test set, it was trained for ten epochs. Still, the KB-BERT failed to perform and obtained an F1-micro and F1-macro of zero. When training the best-achieving baseline classifier, the Decision Trees, on the full training set and testing it on the held-out test set, it achieved an F1-micro of 0.31 and an F1-macro of 0.09.

4.2 ICD Blocks
For the Blocks version of the ICD-10 Corpus, comparing the F1-micro of the KB-BERT with the baseline models during the 10-fold cross-validation, the KB-BERT was superior to the baseline models. The Support Vector Machines was the baseline model with the highest F1-micro. Macro and micro averaged Precision (P), Recall (R), and F1-score (F1) of the KB-BERT and the baseline models during the ten folds are presented in Table 6.
Table 6: Combined scores for the Blocks data set during the 10-fold cross-validation.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Macro</th>
<th>Micro</th>
</tr>
</thead>
<tbody>
<tr>
<td>KB-BERT</td>
<td>0.67</td>
<td>0.55</td>
</tr>
<tr>
<td>SVM</td>
<td>0.76</td>
<td>0.33</td>
</tr>
<tr>
<td>DT</td>
<td>0.54</td>
<td>0.50</td>
</tr>
<tr>
<td>KNN</td>
<td>0.63</td>
<td>0.41</td>
</tr>
</tbody>
</table>

Looking at Figure 4, one can see that, as was the case with the Full codes version of the data set, the classifier ranks comparing the KB-BERT and each baseline classifier remained intact throughout the 10-fold cross-validation for the Blocks version of the data set. This implies that these Wilcoxon test statistics were 0 and that the observed differences between the KB-BERT and the baseline classifiers are likely to reflect that the KB-BERT and the baseline models perform differently on this data set. However, unlike in the case with full codes, the baseline classifiers are not statistically distinguishable.

5 Discussion

5.1 KB-BERT

The results showed that at a block level, the state-of-the-art BERT model trained on Swedish text, KB-BERT, has the potential to be a successful classifier used in an ICD coding tool. It would be interesting to explore if the performance could be further improved by, for example, fine-tuning hyper-parameters such as the activation threshold. It would also be relevant to try other versions of BERT, for instance, a BERT pre-trained on Swedish clinical texts.

While the KB-BERT was the best performing classifier on the block level, it failed to classify full codes. One explanation for this could be that deep learning models are more data-hungry than traditional supervised machine learning models, meaning the KB-BERT suffers the most from moving from ICD blocks to less frequent full codes.

It should also be noted that, like other deep learning models, KB-BERT takes substantially longer to train than the baseline models, resulting in a larger carbon footprint. For example, when considering codes at the block level, it took 300 minutes for a GPU to train (fine-tune) and test the KB-BERT using 10-fold cross-validation. The corresponding number for the slowest baseline model, the Support Vector Machines, was 22 minutes on a regular laptop computer, meaning the actual training time difference probably is even greater than our estimations. While one may argue that the prediction time matters the most, training time might still matter if the idea is that the ICD coding tool should keep learning with time.

5.2 Code Frequencies, Granularity, and Combinations

One finding that stands out is the difference between the classifiers’ performance when considering the 263 full ICD and ICD codes grouped into ten blocks. Comparing the results in Section 4.1 and Section 4.2, the best F₁-micro changes from 0.31 to 0.80 when going from full codes to block codes.

There are several possible explanations for the great difference between the results at a full code level and a block level. Firstly, many of the full codes have very few associated discharge summaries, meaning there are few examples to learn from. Some codes only have one associated discharge summary, leaving no instances to test on,
which leads to $F_1$-scores of zero. In turn, having many low-frequency codes explains the discrepancy between $F_1$-micro and $F_1$-macro scores.

Secondly, as Blanco et al. (2020) suggest, the granularity itself can be a predictor of performance. This means that going from full codes to codes on the block level could have a greater impact than decreasing the number of possible label combinations. Of course, the number of possible label combinations itself also could have impacted the results. One way to address the difficulties associated with ICD coding at the full code level is to combine KB-BERT with the per-label attention mechanism proposed in the article by Blanco et al. (2021).

5.3 Generalisability

One should note that this study’s classifier comparison only is valid for the specific discharge summaries used and that they might not represent Swedish gastrointestinal discharge summaries in general. For example, since the discharge summaries were written between 2007 and 2014, it may be the case that the writing style has changed since. Moreover, the four units that the discharge summaries were created at may not represent Swedish gastrointestinal care units in general.

Furthermore, the results are conditional on the specific instantiations of the classifiers used, and both the KB-BERT and the baseline models may have benefited from hyper-parameter optimisation.

The results of this study are also difficult to compare with the results from other related research since the data sets used differs, and they are often not publicly available because of privacy reasons.

5.4 ICD Coding Tool

This research’s long-term goal is to develop a Swedish ICD coding tool to use in health facilities. Since health personnel assign full codes to the patient records, it would be favourable if the tool suggested full codes. Therefore, it would be suitable for future work to improve the results of this study obtained for codes at the highest level of granularity.

Moreover, it would be interesting to explore how such a tool could incorporate explainability mechanisms. Explainability could be used both as a measure to make the tool trustworthy and to help the coder decide among the suggested codes.

Furthermore, a coding tool would benefit from being developed in close contact with the end-users. Therefore, a design science study with an iterative research approach would be reasonable. In such a study, classifier requirements, such as the desired trade-off between precision and recall, could be discussed.

6 Final Remarks

6.1 Summary

To summarise, the KB-BERT outperformed the baseline classifiers when the full ICD codes of the ICD-10 Corpus were grouped into ten ICD blocks, achieving an $F_1$-micro of 0.80 and an $F_1$-macro of 0.58. These results can be compared to the baseline classifier with the highest $F_1$-micro, the Support Vector Machines, which reached an $F_1$-micro of 0.71 and an $F_1$-macro of 0.42. When considering the 263 full codes, the KB-BERT could not perform at all, obtaining zero $F_1$-micro and $F_1$-macro.

The discrepancy between the results when considering full ICD codes and codes at a block level can partly be because many full codes had very few associated discharge summaries (low frequency). Furthermore, the granularity itself could have impacted the results since distinguishing one full code to another very similar full code is different from distinguishing one ICD block to another, not that similar, ICD block. The fact that one of the tasks had more than 20 times more labels than the other could also have influenced the results.

6.2 Conclusion

In conclusion, this paper contributed to the insufficient knowledge about performing ICD classification on a Swedish corpus by exploring how different classifiers solved a Swedish ICD classification task. One main finding is that KB-BERT showed great potential in predicting few high-frequency ICD codes grouped at the block level. Another main result is that the classifiers, especially the data-hungry KB-BERT, struggled when considering many low-frequency, finely-grained codes.

Since it is desirable for an ICD tool to suggest full codes rather than grouped codes, future work on Swedish ICD classification should focus its efforts on training models that perform well in predicting full codes. One recommendation is to get data from many training examples per each full ICD code to achieve this goal, thereby avoiding the low-frequency issue. Moreover, future stud-
ies would benefit from looking into different ways for classifiers to handle a great amount of finely-grained ICD codes.

Acknowledgments

This work was partially funded by the Norwegian Research Council through the ClinCode project, number 318098.

References


Alberto Blanco, Sonja Remmer, Alicia Pérez, Hercules Dalianis, and Arantza Casillas. 2021. On the contribution of per-ICD attention mechanisms to classify health records in languages with fewer resources than English. In the Proceedings of Recent Advances in Natural Language Processing, RANLP 2021, Varna, Bulgaria.


Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward


Siamese Networks for Inference in Malayalam Language Texts

Sara Renjit
Department of Computer Science
CUSAT, Kochi, India
sararenjit.g@gmail.com

Sumam Mary Idicula
Department of Computer Science
CUSAT, Kochi, India
sumam@cusat.ac.in

Abstract
Natural language inference is a method of finding inferences in language texts. Understanding the meaning of a sentence and its inference is essential in many language processing applications. In this context, we consider the inference problem for a Dravidian language, Malayalam. Siamese networks train the text hypothesis pairs with word embeddings and language agnostic embeddings, and the results are evaluated against classification metrics for binary classification into entailment and contradiction classes. XLM-R embeddings based Siamese architecture using gated recurrent units provides promising results for this classification problem.

1 Introduction
Textual entailment is a uni-directional relationship between two text pairs. It is the method of identifying the meaning from two sentences. Sentence pairs are labeled as entailment pairs if a sentence is inferred from the context of the other sentence in the pair.

It is defined in different ways as classical definition, applied definition, and mathematical definition (Ghuge and Bhattacharya, 2014). In the classical definition, a text entails a hypothesis if the hypothesis is valid in all circumstances where the text is true. In applied definition, text entails hypothesis if the hypothesis is mostly true when a human reads it. The mathematical definition (Glickman et al., 2005) is a text that entails a hypothesis if the probability of the hypothesis is true given the text is greater than the likelihood of the hypothesis being true.

The sentence from which we derive the information is called text, and the sentence which identifies its information as derived from text is called a hypothesis. Sentence pairs are named as entailed if the hypothesis has its meaning derived from the text. Contradictory pairs indicate that the information from the hypothesis is contradictory concerning the information conveyed in the text.

Nowadays, textual entailment has become part of the primary natural language processing tasks. It is also called natural language inference. The terms text and premise are used interchangeably. The performance of the latest transformer-based approaches is evaluated in NLP tasks like entailment recognition, semantic textual similarity, and paraphrase detection. Hence, textual entailment recognition has also become an evaluation criterion for many NLP tasks. It is also a necessary sub-task in applications like multi-document summarization, information retrieval, information extraction, and question answering systems.

Many text entailment or natural language inference related works in English use different sized datasets but not in the Malayalam language. Malayalam is a Dravidian language used in the southern part of India. It is a language that has various dialects and has many inflections. Malayalam language computing is developing, with few resources, namely stemmer, POS tagger, sandhi splitter, and few datasets for paraphrasing, text classification, and sentiment analysis. It has agglutinated language structure, and new words are created through word compounding and inflection, and hence there can be many inferential compound words for a word.

The main challenge of textual entailment recognition in the Malayalam language is the absence of a dataset. Dataset creation is a tedious task that involves high costs and time. We used an in-house dataset created by machines and humans in the loop translation of the Stanford Natural Language Inference dataset. It is a very cost-effective method of dataset creation that can be adapted to any low-resource language. This work attempts to use Siamese networks with bidirectional long
short term memory and gated recurrent units to understand the similarities and differences of text-hypothesis pairs for classification.

The remaining sections are organized as follows. Section 2 briefs the works related to textual entailment recognition. Section 3 details the dataset. Section 4 discusses the design of the system for classification. The experimental settings, results, and discusses sentence similarity are in Section 5. Section 6 concludes the work.

2 Related Works

Textual entailment recognition started with a Recognizing Textual Entailment (RTE) challenge in 2005 (Dagan et al., 2005). This challenge continued for years and used different-sized datasets. As the dataset size increases, feature-based methods such as word overlap, n-gram match, set-based similarities, and syntactic similarities were replaced by machine learning and deep learning methods using different word and sentence representations.

There are numerous entailment recognition systems in English and other languages like French, German, Italian, Spanish, and Arabic. Various textual entailment frameworks have also been developed in these languages, namely EXCITEMENT Open Platform (EOP) (Magnini et al., 2014). EOP uses edit distance and classification as its algorithms. Lexical level features, syntactic and surface-level features, graph-based approaches were used to recognize entailments. The increase in dataset size has helped to use machine learning and deep learning strategies for this classification. Deep learning methods are widely used for textual entailment in English using different datasets.

SNLI (Stanford Natural Language Inference) dataset is a collection of 570k sentence pairs mainly collected through Amazon Mechanical Turk, referencing the Flickr30k corpus. This dataset helped in using deep learning techniques to text entailment recognition. Sentence models with the sum of words, recurrent neural networks, and long short-term memory networks were discussed (Bowman et al., 2015).

MNLI (Multi-Genre Natural Language Inference) dataset is a collection of 433k sentence pairs from different written and spoken English genres. Some of the genres are face-to-face, government, telephone, letters, fiction, and travel. It is an improvement from SNLI with a more diverse collection of sentence pairs, and hence its baseline performance is low compared to SNLI (Williams et al., 2018).

XNLI (Conneau et al., 2018) (Cross-Lingual Natural Language Inference) corpus is a collection of data from MNLI, derived for 15 languages, including some low resource languages like Urdu and Swahili. It uses a translation-based approach with multilingual sentence encoders and then aligning sentence embeddings for inference identification. Except for English, entailment recognition systems exist for French, Spanish, German, Greek, Bulgarian, Russian, Turkish, Arabic, Vietnamese, Thai, Chinese, Hindi, Swahili, and Urdu. In languages like Swahili, Thai and Urdu, transfer learning based approaches are used, which is helpful for small-sized datasets.

For the Malayalam language, there are works related to paraphrasing, sentiment analysis, summarization, whereas text entailment recognition is seemingly a new area for Malayalam language processing. The performance of different embedding-based approaches is one work in this language for entailment recognition (Renjit and Idicula, 2021). In this work, LASER-based embedding representation showed improved performance results for entailment recognition for the Malayalam language compared with BERT and other models.

Bidirectional LSTM based dependent reading represents text and hypothesis in encoding and inference stage (Ghaeini et al., 2018). Siamese network-based architecture with sentence embeddings from BERT experiments in the English language (Reimers et al., 2019). A neural model based on LSTM using the word by word attention is another deep learning-based method for recognizing entailments (Rocktäschel et al., 2015). Child-Sum-Tree-based inference of texts generalizes well for SNLI and other entailment datasets (John et al., 2016). Text alignment based approaches along with machine learning are used for entailment recognition in Arabic language (Boudaa et al., 2019). Another method used asymmetric word embeddings to produce similarity based word-word interactions for textual entailment (Ma et al., 2018).

Entailment recognition has been part of Competition on Legal Information Extraction / Entailment, where sentence encoding and decomposable attention models perform entailment recognition in the context of legal texts (Son et al., 2017). Automatic translation-based approaches are used in the Italian dataset, where the dataset is translated into the En-
English language for entailment recognition (Pakray et al., 2012).

3 Dataset

The subset of the Malayalam Language Inference (MaNLI) dataset is used for binary entailment recognition. It consists of 7989 text hypothesis pairs which are labeled as Entailment and Contradiction. This dataset is created from Stanford Natural Language Inference (SNLI) dataset. The sentence pairs from the SNLI dataset are translated to the Malayalam language with linguistic corrections from the Department of Linguistics, Malayalam University, Kerala.

There are 4026 entailment pairs and 3963 contradiction pairs in this dataset. The reason for the creation of this dataset is the unavailability of entailment datasets in Malayalam. Languages like Malayalam have many inherent linguistic properties like inflections, agglutinative nature, dialect-related differences, and no specific word order.

A sample from the dataset is shown in Figure 1.

Figure 1: Sample dataset

The English translation of the sample dataset is provided in Table 1.

<table>
<thead>
<tr>
<th>Premise</th>
<th>Hypothesis</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two men on bicycles.</td>
<td>People are riding bikes.</td>
<td>Entailment</td>
</tr>
<tr>
<td>Two men on bicycles.</td>
<td>A few people are catching fish.</td>
<td>Contradiction</td>
</tr>
</tbody>
</table>

Table 1: Sample Dataset translation in English

4 Proposed Architecture

The design of this system consists of neural networks of identical architecture consisting of an embedding layer and bidirectional long short-term memory/gated recurrent unit networks. The Siamese network for sentence representation takes each sentence from text-hypothesis pair to an embedding layer. This layer uses different types of embeddings, namely Word2Vec and LASER (Language Agnostic Sentence Representations), and XLM-R. The different layers in the system are:

4.1 Embedding

The first layer is the embedding layer, where each sentence in text or hypothesis gets an efficient representation that captures its meaning in high dimensional vector space. In this layer, we used approaches, namely, Word2Vec, language-agnostic sentence representation (LASER), and XLM-R.

4.1.1 Word2Vec

Word2vec (Mikolov et al., 2013) is a word embedding neural model that produces distributed representation of words in vector space. The neural model trains in two ways, namely skip-gram and continuous bag of words, using hierarchical softmax or negative sampling (Rong, 2014). Words having semantic similarity represented through vectors are closer in high dimensional space. The embeddings from this model are input to the Keras embedding layer in the form of an embedding matrix to obtain text representation and hypothesis.

4.1.2 LASER

Language Agnostic Sentence Representations (Artetxe and Schwenk, 2019) is a toolkit modeled for more than 90 languages, including the Malayalam language. LASER embeddings are representations of sentences so that a sentence representation in two or more languages will be close to each other in their high dimensional vector space. It also uses an encoder-decoder architecture based on neural machine translation.

4.1.3 XLM-R

XLM-R (Conneau et al., 2019) is a self-supervised model that is trained on cross-lingual representations. This transformer-based masked language model is trained for 100 languages, including the Malayalam language. The cross-lingual sentences are taken from Common Crawl data.

4.1.4 Dimensionality Reduction using PCA

Principal component analysis (PCA) reduces the sentence embedding obtained from the LASER model. It compromises with accuracy, and hence selecting an adequate number of features is critical to the model. Depending on the system’s configuration, dimensions of 100, 500, and 1000 are tested for 1024 dimensional LASER embeddings. Dimension reduction of 100 leads to 0.87% loss
of features, dimension 500 causes 0.488% feature loss, and dimension 1000 led to 0.023% feature loss.

4.2 BLSTM
The bidirectional LSTM layer consists of two LSTM layers, one in forward and the other in the backward direction for sequence processing. It is used when a sequence of data is to be processed. The text and hypothesis are passed through BLSTM layers to obtain a sequence representation that embeds the complete information.

4.3 GRU
This layer has recurrent neural networks with a gating mechanism. It is similar to long short-term memory networks, and it does not have output gates. As such, it has less number of parameters with good performance on small-sized datasets.

4.4 CONCATENATION
The sequence representation of text and hypothesis are then concatenated in this layer, and batch normalization is done. Dropout configurations are then applied and passed to a Dense layer.

4.5 DENSE LAYER
The dense layer has its input from the concatenation layer and has rectified leaky unit activation function. It is then batch normalized, and dropout is applied and flattened to feed to the following dense layer.

4.6 SIGMOID CLASSIFICATION
For binary classification, the sigmoid activation function is used in the final dense layer. The sigmoid function is given by

\[ S(x) = \frac{1}{1 + e^{-x}} \]  

(1)

The system design is shown in Figure 2. The representations from the embedding layer is then passed to a bidirectional LSTM / GRU layer where each sentence gets a context representation. It is performed for both text and hypothesis in Siamese network architecture, followed by a concatenation of the outputs from BiLSTM / GRU. The concatenated text hypothesis representation is then fed to a Dense layer with ‘RELU’ activation followed by classification. Sigmoid function with binary cross-entropy loss function performs the model training and classification.

5 Experimental Results

5.1 Experimental Setup
Implementations used Google Colab platform and Spyder IDE using Python, Tensorflow, and Keras library for machine learning and Scikit-Learn for evaluations.

The parameter configurations for training the system are 100 LSTM nodes, 100 dense units, RELU activation function for dense layer, drop-out rate of 0.17, 0.25 for LSTM and 0.25 dropout for dense layer.

5.2 Results
The results are evaluated in terms of classification metrics, namely Precision(P), Accuracy, Recall(R), F1-score(F1), and Support(S). Experimental results of the Siamese network architecture detailed above are shown in Table 2.

<table>
<thead>
<tr>
<th>Class</th>
<th>P</th>
<th>R</th>
<th>F1</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contradiction</td>
<td>0.71</td>
<td>0.60</td>
<td>0.65</td>
<td>500</td>
</tr>
<tr>
<td>Entailment</td>
<td>0.65</td>
<td>0.76</td>
<td>0.70</td>
<td>500</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.68</td>
<td>0.68</td>
<td>0.68</td>
<td>1000</td>
</tr>
<tr>
<td>Macro average</td>
<td>0.68</td>
<td>0.68</td>
<td>0.68</td>
<td>1000</td>
</tr>
<tr>
<td>Weighted average</td>
<td>0.68</td>
<td>0.68</td>
<td>0.68</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 2: Results based on LASER embedding with dimensions reduced to 1000.

From Table 3, we observe that reducing the embedding dimension to 500 through the principal
component analysis technique results in performance drops, as there is information loss.

<table>
<thead>
<tr>
<th>Class</th>
<th>P</th>
<th>R</th>
<th>F1</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contradiction</td>
<td>0.91</td>
<td>0.19</td>
<td>0.31</td>
<td>500</td>
</tr>
<tr>
<td>Entailment</td>
<td>0.55</td>
<td>0.98</td>
<td>0.70</td>
<td>500</td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td>0.58</td>
<td>1000</td>
</tr>
<tr>
<td>Macro average</td>
<td>0.73</td>
<td>0.58</td>
<td>0.51</td>
<td>1000</td>
</tr>
<tr>
<td>Weighted average</td>
<td>0.73</td>
<td>0.58</td>
<td>0.51</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 3: Results based on LASER embeddings with dimensions reduced to 500.

When the embedding dimension is reduced to 100, we obtain the results in Table 4. Hence dimensions of 100 and 1000 yield good results compared with dimension 500. It resulted due to a mismatch in network configuration with embedding size.

<table>
<thead>
<tr>
<th>Class</th>
<th>P</th>
<th>R</th>
<th>F1</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contradiction</td>
<td>0.68</td>
<td>0.62</td>
<td>0.65</td>
<td>500</td>
</tr>
<tr>
<td>Entailment</td>
<td>0.65</td>
<td>0.71</td>
<td>0.68</td>
<td>500</td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td>0.66</td>
<td>1000</td>
</tr>
<tr>
<td>Macro average</td>
<td>0.66</td>
<td>0.66</td>
<td>0.66</td>
<td>1000</td>
</tr>
<tr>
<td>Weighted average</td>
<td>0.66</td>
<td>0.66</td>
<td>0.66</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 4: Results based on LASER embeddings with dimension reduced to 100.

**Comparison with word2vec:** The system design is compared with word2vec based embedding. The dataset is trained using the Word2vec model with dimension 100, minimum count of words 1. The difference in configurations showed better results, as shown in the tables below.

**Configuration 1:** With negative sampling and using a continuous bag of words approach for Word2Vec model produced the results as in Table 5.

<table>
<thead>
<tr>
<th>Class</th>
<th>P</th>
<th>R</th>
<th>F1</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contradiction</td>
<td>0.60</td>
<td>0.43</td>
<td>0.50</td>
<td>500</td>
</tr>
<tr>
<td>Entailment</td>
<td>0.56</td>
<td>0.71</td>
<td>0.63</td>
<td>500</td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td>0.57</td>
<td>1000</td>
</tr>
<tr>
<td>Macro average</td>
<td>0.58</td>
<td>0.57</td>
<td>0.57</td>
<td>1000</td>
</tr>
<tr>
<td>Weighted average</td>
<td>0.58</td>
<td>0.57</td>
<td>0.57</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 5: Results based on Word2Vec with Configuration 1.

**Configuration 2:** With hierarchical softmax and skip-gram based approach resulted in Table 6. We infer that Word2Vec of 100 dimensions with hierarchical softmax and LASER embedding reduced to 1000 dimension shows good performance. Hence Word2vec is better for this Siamese network based architecture based on its performance with lesser dimensional embeddings.

**Comparison with GRU** For the same architecture, when BiLSTM is replaced with GRU, Word2Vec based system showed the same performance as below.

<table>
<thead>
<tr>
<th>Class</th>
<th>P</th>
<th>R</th>
<th>F1</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contradiction</td>
<td>0.76</td>
<td>0.51</td>
<td>0.61</td>
<td>500</td>
</tr>
<tr>
<td>Entailment</td>
<td>0.63</td>
<td>0.84</td>
<td>0.72</td>
<td>500</td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td>0.67</td>
<td>1000</td>
</tr>
<tr>
<td>Macro average</td>
<td>0.69</td>
<td>0.67</td>
<td>0.66</td>
<td>1000</td>
</tr>
<tr>
<td>Weighted average</td>
<td>0.69</td>
<td>0.67</td>
<td>0.66</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 7: Results based on Word2Vec with GRU layer instead of BiLSTM

**LASER based GRU system shows the below results for classification as in Table 8.**

<table>
<thead>
<tr>
<th>Class</th>
<th>P</th>
<th>R</th>
<th>F1</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contradiction</td>
<td>0.73</td>
<td>0.29</td>
<td>0.42</td>
<td>500</td>
</tr>
<tr>
<td>Entailment</td>
<td>0.56</td>
<td>0.89</td>
<td>0.69</td>
<td>500</td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td>0.59</td>
<td>1000</td>
</tr>
<tr>
<td>Macro average</td>
<td>0.64</td>
<td>0.59</td>
<td>0.55</td>
<td>1000</td>
</tr>
<tr>
<td>Weighted average</td>
<td>0.64</td>
<td>0.59</td>
<td>0.55</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 8: Results based on LASER with GRU layer instead of BiLSTM

**Comparison with XLM-R embeddings** XLM-R is a masked language model trained for 100 languages, including Malayalam. This transformer-based architecture produced the results shown in Table 9. The default dimension is 768, which is reduced to 100 dimensions.

Table 9 shows the accuracy values obtained for different configurations of Siamese networks. It

<table>
<thead>
<tr>
<th>Class</th>
<th>P</th>
<th>R</th>
<th>F1</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contradiction</td>
<td>0.76</td>
<td>0.49</td>
<td>0.60</td>
<td>500</td>
</tr>
<tr>
<td>Entailment</td>
<td>0.63</td>
<td>0.85</td>
<td>0.72</td>
<td>500</td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td>0.67</td>
<td>1000</td>
</tr>
<tr>
<td>Macro average</td>
<td>0.69</td>
<td>0.67</td>
<td>0.66</td>
<td>1000</td>
</tr>
<tr>
<td>Weighted average</td>
<td>0.69</td>
<td>0.67</td>
<td>0.66</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 6: Results based on Word2Vec with Configuration 2.
5.3 Sentence Similarity

As part of this classification, similarities of sentences with respect to text hypothesis pairs are measured quantitatively as entailment confidence. For example, the similarity score obtained for the classification of instances is shown in Figure 3. The entailment confidence score is helpful for sentence similarity tasks to identify the extent to which the pairs are similar. Thus it also aids in multidocument summarization tasks, in which we can avoid similar sentences in summary based on the entailment/similarity score.

6 Conclusion

In this work, we focused on the application of Siamese network architecture to recognize entailment in Malayalam language. The results show adequate performance. The use of newer embedding models leads to better accuracy but the embedding dimension is a limiting factor with the network configuration. As the embedding dimension increases, the time and space complexity increases in Siamese model architecture, where text and hypothesis are processed as sequences parallely.

Through this work, we aim to depict the performance of Siamese network based entailment recognition with respect to Malayalam language, which is a low resource Dravidian language.

References


A Call for Clarity in Contemporary Authorship Attribution Evaluation

Allen Riddell
Indiana University Bloomington
Bloomington, Indiana, USA
riddella@indiana.edu

Haining Wang
Indiana University Bloomington
Bloomington, Indiana, USA
hw56@indiana.edu

Patrick Juola
EVL Lab, Duquesne University
Pittsburgh, Pennsylvania, USA
juola@mathcs.duq.edu

Abstract
Recent research has documented that results reported in frequently-cited authorship attribution papers are difficult to reproduce. Inaccessible code and data are often proposed as factors which block successful reproductions. Even when original materials are available, problems remain which prevent researchers from comparing the effectiveness of different methods. To solve the remaining problems—the lack of fixed test sets and the use of inappropriately homogeneous corpora—our paper contributes materials for five closed-set authorship identification experiments. The five experiments feature texts from 106 distinct authors. Experiments involve a range of contemporary non-fiction American English prose. These experiments provide the foundation for comparable and reproducible authorship attribution research involving contemporary writing.

1 Introduction
Closed-set authorship attribution picks out the likely author of an unsigned document from a pool of candidate authors. Decades of research show that authors leave conspicuous “fingerprints” in their writing (Juola, 2006). A small amount of pre-existing prose (ca. 2,500 words) is often enough to learn enough about the writing “styles” of a set of candidate authors to correctly identify the author of an unsigned document. Authorship attribution techniques have found application in numerous domains. They have been used to resolve uncertainty about authorship in historical research (Mosteller and Wallace, 1964). For writers living today, widespread use of authorship attribution techniques—and related author profiling techniques (Argamon et al., 2009)—poses a privacy risk (Brennan et al., 2012). A better understanding of how authorship attribution techniques work can inform efforts to improve privacy-enhancing language technologies.

While there is no doubt that authorship attribution methods have improved over the past century, recent progress is harder to measure. Some of this difficulty is due to the field’s success. In recent decades, new methods improve on old ones by small amounts. Given small improvements, assessing whether or not the advance may be due to aleatory factors such as preprocessing or a particular dataset becomes difficult. Another reason recent progress is difficult to measure is the lack of standard benchmark tasks. Of 15 frequently-cited authorship attribution studies examined by Potthast et al. (2016), original corpora could be found for only 4 (27%) and code could be located for 0 (0%). While other fields, notably machine translation and language modeling, excel at organizing research activity around publicly-accessible benchmark tasks, contemporary authorship attribution research has no such tasks.

Recent experience suggests that without standard benchmarks—and evidence that researchers can consistently reproduce results using them—a field’s ability to self-assess progress on well-defined tasks can go astray. The field of recommender systems offers a cautionary tale. Rendle et al. (2019) documents a series of papers being published in prestigious journals over a five year period which do not, in fact, improve on earlier results. Notably, these papers used a standard dataset for their evaluations (Movielens 10M). Where these papers fell short was in their reproduction of previous results—to which their new methods were compared. The papers reported improvements on earlier results...
which were illusory; models used in earlier research, upon closer examination, outperformed the new methods. Analogous cases exist in other fields. In machine translation, although standard datasets were used, inconsistency in applying a key metric (BLEU) prevented researchers from easily reproducing or comparing results (Post, 2018).

Our paper supports reproducible research in authorship identification by introducing five standard benchmark tasks. Each task features fixed train and test sets. Four of the five tasks have a test set consisting of writing samples on fixed topics, guaranteeing that test set examples do not overlap with training set examples in terms of subject matter. Data for all tasks is available for download without any restrictions.

2 Problem Description

2.1 Problem: Models Cannot be Compared due to Unavailable or Under-specified Test Sets

Comparing the effectiveness of a new model with that of an existing model requires, at minimum, evaluating models on the same data. Because different models may perform differently when applied to texts by different authors or to texts in different genres by the same authors, comparing the performance of two models on a new dataset is often uninformative. Even when the new dataset resembles the original, researchers should worry that the poor performance of an earlier model may be due to accidental errors in re-implementation. Reliable comparisons of new models with previous methods are, in fact, picking up on content-independent authorial fingerprints, test set texts should not resemble training set texts.

For an illustration of the problem, consider the use of a corpus of 100 newspaper articles written by 10 different authors. Using such a corpus to evaluate the performance of an authorship attribution method may not yield the expected information: an estimate of how well the method will perform on similar authors in a different setting. The risk of a model using topical information is clear. Newspaper writers tend to have distinct areas of expertise (“beats”) which influence the types of subjects they write about. Writers from the same generation or similar social backgrounds may tend to write about certain topics. Senior writers may be more likely than junior writers to receive certain topics as assignments. Methods which appear to be using content-independent features may, in fact, be picking up on subtle signals of topic.

Unfortunately this kind of homogeneity in evaluation corpora is common. It features in all the corpora considered by Abbasi and Chen (2008) as well as the “C10” corpus drawn from Reuters-RCV1 (Potthast et al., 2016).

One method of addressing this problem is to use test set documents which are distinct from training set documents. Test set documents might be written in a different setting or different document genre. If, say, training set documents are work e-mails, then test set documents might be personal essays. Using test documents from a different time period would also help address the concern of topical homogeneity. Koppel et al. (2009) illustrate such a
division in a dataset involving two authors by using e-mails written before a fixed date as training and e-mails written after the date as testing.

Another method involves conducting a field experiment and eliciting prose on a fixed topic from writers. The elicited writing samples form the test set. This method is expensive but guarantees that models will not perform better by leveraging information about the topics specific authors tend to write about. Both Juola (2004) and Brennan et al. (2012) use this approach.

Authorship attribution methods are consistently presented as relying on the identification of topic-independent fingerprints. Evaluation tasks should be aligned with this presentation.

2.3 Problem: Unavailable or Restricted Corpora

The practice of restricting access to corpora appears to be more common in authorship attribution research than in the machine translation and language modeling communities. We considered including the C10, PAN12, and PAN13 authorship attribution tasks in our suite of benchmark tasks but found that all three are restricted and cannot be downloaded without permission.1 We know of no cases in current machine translation or language modeling research where performing a standard evaluation requires access to a restricted dataset. Data for the news translation tasks distributed by the Conference on Machine Translation are available for immediate download.2 Data for the widely-used language modeling benchmarks (GLUE, SQuAD) are publicly available (Wang et al., 2018; Rajpurkar et al., 2018). Of the 81 language modeling tasks cataloged by the NYU-based team developing the Jiant evaluation tool, 69 tasks (85%) can be downloaded automatically, that is, by the evaluation software itself.3

Making a dataset publicly available increases the likelihood that other researchers will reproduce results. Recent experience has shown that the probability that a result may not be reproducible or replicable is higher than previously appreciated (less than 70% according to Baker (2016)). The problem of non-reproducible results is sufficiently serious that certain conferences are exploring adopting additional measures—beyond submission of code and data—which will alleviate the problem.4

There is no reason to suspect that the reproducibility rate of authorship attribution research is conspicuously different from the rate in other areas of computational linguistics. Indeed, in the study of 15 frequently-cited authorship attribution papers, Potthast et al. (2016) document one failure to replicate results (Potthast et al., 2016, 403). If reproducing or replicating results is difficult in as many as 6% (1 in 15) of papers, then reproduction (or replication) should be a be a regular practice. And reproducing results requires that the original code and data be easy to access.

3 Improving Authorship Attribution Evaluation

The problems described in the previous section complicate a range of authorship attribution research (e.g., identification, verification, profiling). We propose a suite of five tasks which address the problems for one area of authorship attribution research: closed-set author identification involving contemporary English-language non-fiction prose. Lessons learned developing standard benchmark tasks in this area will, we hope, inform the development of analogous tasks in other areas.

Two arguments support our focus on contemporary non-fiction texts. First, collecting redistributable non-fiction prose from a diverse set of writers is relatively easy. A considerable share of the English-using population writes non-fiction prose. Demonstrating (some) competency in English composition is a requirement in secondary education across the English-speaking world. Second, many researchers are interested in the efficacy of authorship attribution methods applied to contemporary non-fiction English prose. English is, for the moment, the lingua franca of diplomacy, science, and international business. Authorship attribution methods which work on English therefore enjoy broad applicability. The stakes of author profiling research—research informed by authorship attribution research—are also significantly higher for research involving living writers than for writ-

---

1The restricted-download datasets may be found at the following URLs: https://zenodo.org/record/3759064 (C10), https://zenodo.org/record/3713273 (PAN12), https://zenodo.org/record/3715864 (PAN13).
2For example, http://www.statmt.org/wmt18/translation-task.html
3See https://github.com/nyu-mll/jiant/blob/master/guides/tasks/supported_tasks.md for a list of the tasks.
ers active in previous centuries. Only in the former case is, say, an individual’s privacy at risk.

3.1 Reproducible Authorship Attribution Benchmark Tasks (RAABT)

Five closed-set authorship identification tasks make up the Reproducible Authorship Attribution Benchmark Tasks (RAABT). Table 1 summarizes the tasks. All tasks feature a fixed test set. Test set documents do not overlap with training set documents. In four out of five of the tasks, authors write test set documents on a fixed topic. Three of the tasks involve writing from a diverse set of adults living in North America. In aggregate, the tasks feature 106 different authors.

The tasks are published at [https://zenodo.org/record/5213898](https://zenodo.org/record/5213898).

<table>
<thead>
<tr>
<th>Task</th>
<th>Number of authors (training set)</th>
<th>Words per author (training set)</th>
<th>Number of authors (test set)</th>
<th>Words per author (test set)</th>
<th>Fixed topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAC–fixed-topic</td>
<td>13</td>
<td>2,563</td>
<td>13</td>
<td>843</td>
<td>Yes</td>
</tr>
<tr>
<td>AAAC–free-topic</td>
<td>13</td>
<td>2,563</td>
<td>13</td>
<td>2,008</td>
<td>No</td>
</tr>
<tr>
<td>BGE–obfuscation</td>
<td>45</td>
<td>8,866</td>
<td>45</td>
<td>555</td>
<td>Yes</td>
</tr>
<tr>
<td>RJ–fixed-topic</td>
<td>48</td>
<td>7,492</td>
<td>21</td>
<td>575</td>
<td>Yes</td>
</tr>
<tr>
<td>RJ–obfuscation</td>
<td>48</td>
<td>7,492</td>
<td>27</td>
<td>565</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1: Summary statistics of documents used in the five tasks.

3.2 Task descriptions

1. Ad-hoc Authorship Attribution Competition, fixed topic (AAAC–fixed-topic). The first task is “Problem A” from the 2004 Ad-hoc Authorship Attribution Competition (AAAC) ([Juola, 2004, 2006]). Texts were gathered from 13 authors in 2013 undergraduate writing course at a university in the United States. For the test set documents, participants were asked to write on the topic of “work.”

2. Ad-hoc Authorship Attribution Competition, free topic (AAAC–free-topic) The second task is “Problem B” from the AAAC. Test documents are additional course essays on other topics. Test set documents do not overlap with training documents. Training documents are the same as in the first task.

3. Extended Brennan-Greenstadt Corpus, obfuscation condition (EBG–obfuscation) The Extended Brennan-Greenstadt Corpus ([Brennan et al., 2012]) (EBG) contains writing from 45 individuals contacted through the Amazon Mechanical Turk platform no later than the year 2012. Participants uploaded examples of their writing. The researchers asked for writing of a “scholarly” nature. Participants were then asked to write a short essay on a fixed topic. They were asked to describe their neighborhood to someone unfamiliar with the location. Notably, they were also asked to obscure their writing style. They were, however, not given any instructions on how to accomplish this. These essays form the test set.

Given prevailing norms on Amazon Mechanical Turk and the monetary incentive to finish quickly (payment did not depend on time spent on the task) we suspect many participants did not devote considerable time to de-
vising strategies for obscuring their writing style. We suggest that this task be treated as, in essence, an additional fixed topic task.

We note that the population of individuals who sell their labor on Amazon Mechanical Turk is quite diverse in terms of age, gender, and region (Coppock and McClellan, 2019).

4. **Riddell-Juola Corpus, control condition (RJ-fixed-topic)**

The Riddell-Juola Corpus collects texts using essentially the same techniques were used in Brennan et al. (2012). Responses were collected in March and June of 2019. According to self-reported gender and age, participant demographic characteristics are roughly balanced.

Participants were asked to respond to the same “describe your neighborhood” prompt mentioned earlier. No further instructions were given. (The instruction to obscure one’s writing style was not present.)

5. **Riddell-Juola Corpus, obfuscation condition (RJ-obfuscation)**

This task is the same as RJ-fixed-topic with one difference. Participants were told to obscure their writing using the same instruction as found in EBG-obfuscation. Again, they were given no instructions on how to accomplish this task.

Participants were randomly assigned to receive the obfuscation instruction. Therefore the authors of the test set documents in this task do not overlap with the authors of the test set documents in RJ-fixed-topic.

The training sets for the two tasks involving the Riddell-Juola Corpus are the same.

4 Accuracy of Received Methods

Table 2 reports the performance of two classic methods on the five tasks. We use a familiar 512-word function word feature set with both methods (Koppel et al., 2009). For linear SVM we use the libSVM implementation with default cost parameter \( C = 1 \) (Chang and Lin, 2011). For multiclass logistic regression we use L2 regularization \( \lambda = 1 \) (Pedregosa et al., 2011).

These baselines are intended to be reference points. They are chosen because they should be particularly easy to reproduce.

5 Discussion

Perceptions of the importance of having reproducible measures of model performance on well-understood tasks have changed over the last decade. Previously regarded as something desirable but by no means essential, reproducible benchmarks are increasingly seen as indispensable. Experience has shown that without such benchmarks, researchers risk overestimating the reliability of existing results or gaining a false sense of a field’s progress on particular problems. Our paper contributes a suite of benchmarks which can be used to anchor future authorship attribution research.

These five tasks are a start. Additional tasks would be welcome. Many forms of writing and document types in widespread use today are not featured in the five tasks we introduce here. Short text messages and informal e-mails, in particular, are ubiquitous. Yet many individuals’ habits of composition vary dramatically when writing in such genres. Standard benchmarks for cross-register and cross-genre authorship attribution would likely yield new insights into the strengths and weaknesses of existing approaches.

Acknowledgments

This work was supported by the National Science Foundation under Grant No. 1814425. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References


Michael Brennan, Sadia Afroz, and Rachel Greenstadt. 2012. Adversarial Stylometry: Circumventing Authorship Recognition to Preserve Privacy


Frederick Mosteller and David L Wallace. 1964. *Inference and Disputed Authorship: The Federalist*. Addison-Wesley, Reading, MA.


Varieties of Plain Language

Allen Riddell
Indiana University Bloomington
Bloomington, Indiana, USA
riddella@indiana.edu

Yohei Igarashi
University of Connecticut
Storrs, Connecticut, USA
yohei.igarashi@uconn.edu

Abstract

Many organizations seek or need to produce documents that are written plainly. In the United States, the “Plain Writing Act of 2010” requires that many federal agencies’ documents for the public are written in plain English. In particular, the government’s Plain Language Action and Information Network (“PLAIN”) recommends that writers use short sentences and everyday words, as does the Securities and Exchange Commission’s “Plain English Rule.” Since the 1970s, American plain language advocates have moved away from readability measures and favored usability testing and document design considerations. But in this paper we use quantitative measures of sentence length and word difficulty that (1) reveal stylistic variation among PLAIN’s exemplars of plain writing, and (2) help us position PLAIN’s exemplars relative to documents written in other kinds of accessible English (e.g., The New York Times, Voice of America Special English, and Wikipedia) and one academic document likely to be perceived as difficult. Uncombined measures for sentences and vocabulary—left separate, unlike in traditional readability formulas—can complement usability testing and document design considerations, and advance knowledge about different types of plainer English.

1 Introduction

The quality of being “plain” has been held up as a stylistic ideal in English prose since the later seventeenth century (Guillory, 2017). This ideal has shown remarkable persistence (Cutts, 2020). In the United States, the plain language movement took off in the 1940s, and plainness remains a stylistic goal for many kinds of organizations in their writing on websites and other publications: medical and public health information, insurance policies, instructions to jurors, loan agreements, and Social Security benefits statements, to name just a few (Schriver, 2017; Cutts, 2020).

Since the passage of the Plain Writing Act of 2010, American federal agencies must use “plain language,” defined as “writing that is clear, concise, well-organized, and follows other best practices appropriate to the subject or field and intended audience” (United States Congress, 2010). Affecting all 2.1 million employees of the U.S. Federal government (Jennings and Nagel, 2020), the Act requires that agencies use this kind of language in many of their documents for the public, train their employees in this style, and demonstrate their compliance with the Act.1 There are several cogent rationales for plain language use: it grants access to understandable information to a greater number of people with differing literacy levels; it saves agencies the labor and money involved in clearing up confusing communications; and it may help to restore citizens’ trust in public-serving organizations, an especially important agenda in our era of misinformation, disinformation, and propaganda (Schriver, 2017). If we add to the U.S. government’s plain writing imperative the similarly simple writing style espoused by Big Tech in all of its apps, websites, and documentation, we can see that plain language is a dominant discursive goal today.2

Although our focus is on the American context, it is worth noting that plain English is pursued globally. Another U.S. government plain language mandate, the Securities and Exchange Commission’s plain writing initiatives of 1998 and 2008 not only extend plain language into the private sector but also include the requirement that foreign

---

1For an example of the last, see the Department of Health and Human Services’ 2021 Plain Writing Act Compliance Report.

2See, for example, the Microsoft Writing Style Guide and the Google Developer Documentation Style Guide.
firms listing shares on U.S. stock exchanges use plain English in their prospectuses (SEC, 2021). There is also the Canada-based organization, Plain Language Association International (International, 2021), as well as one based in the UK, Clarity, which focuses on making legalese plainer (Clarity, 2021). Plain writing is nowhere—it is supposed to be inconspicuous writing that functions like a transparent medium for information—and everywhere.

Quantitative measures of plainness are generally out of favor these days among plain English advocates. Early on, the American plain language movement was associated with readability formulas: most notably the Flesch-Kincaid formula (still available in Microsoft Word), the similar Dale-Chall formula, and the Gunning fog index (Klare, 1963). But since the late 1970s, plain language advocates have adopted usability testing and emphasized design considerations beyond words and sentences: that is, information, document, and visual design (Redish, 2000; Schriver, 2017). We enumerate the main limitations of existing readability formulas below.

At the same time, there are quantifiable features of plain writing. The American government-advocacy network, the Plain Language Action and Information Network (“PLAIN”), includes among its techniques for writers the use of “short sentences” and “common, everyday words” (PLAIN, 2021). Similarly, the SEC’s “Plain English Rule” is defined by six principles, the first two of which are “short sentences” and “definite, concrete, everyday language” (SEC, 1998b). And the Oxford Guide to Plain English’s first two guidelines that pertain to style—the guidelines immediately after “Plan before you write” and “Organize your material...”—concern sentences and words: “Over the whole document, make the average sentence length 15-20 words;” and “Use words your readers are likely to understand” (Cutts, 2020).

In this paper, we use quantitative measures of sentence length and word difficulty to evaluate some of the documents identified on the PLAIN website as good models of plain writing. But rather than combining sentence measures and word measures into a single readability score—a single score that is of little use to individual writers trying to make their writing plainer—we take the simple step of keeping each measure separate; we entertain the possibility that disjoined measures might be more illuminating and helpful to writers. We find, for example, that a text belonging to a domain (academic philosophy) oftentimes charged with jargon, does not use extremely difficult words, although it does have long sentences. Using these two separate measures, we reveal among PLAIN’s exemplars a degree of variation that complicates current understandings of plain writing. Furthermore, in order to understand better what plainness is in all of its variety, we position PLAIN’s different exemplary documents in relation to other documents written in other kinds of relatively accessible English (The New York Times, Voice of America Special English, and Wikipedia) and one academic document belonging to a genre perceived to be difficult (mentioned above). We propose that quantitative measures complement current approaches to plain writing and that advancing our knowledge about plain writing will require a better sense of the different types of plainer English.

2 The Problem with Existing Readability Measures

Plain language advocates have described several limitations to classic readability measures from the mid-twentieth century, including the Flesch-Kincaid formula. These measures, originally designed to measure children’s reading abilities, do not accurately measure the reading abilities of adult information consumers. They also generate one-size-fits-all scores regardless of audience; worse, these scores do not typically give writers helpful guidance toward improving a piece of writing through revision. Most of all, for modern proponents of plain writing, readability formulas fail to take into account all of the non-prose elements of websites and other documents. These include the organization of information, the use of headings, tables of contents, layout and formatting, visuals, and so on (Redish, 2000; Redish and Selzer, 1985). Therefore, although readability measures have in the past been used to evaluate the accessibility of government documents, such measures are not mentioned in either the Plain Writing Act of 2010 or the SEC’s Plain English Rule of 1998.

Existing quantitative approaches have additional limitations. They prove brittle in practice: most formulas measure a word’s accessibility using the word’s length in syllables or characters. The problem with this method is that many common words are long and many rare words are short: “international,” “communication,” “relationship,” and “en-
tertainment” are far more likely to be understood than “mien,” “feign,” “pang,” “dote,” and “cinch.”

The methods are also vulnerable to gaming (Redish, 2000). For example, Flesch-Kincaid readability scores can be raised by replacing lengthy words with acronyms. For example, an Internal Revenue Service (IRS) form designer describes replacing “self-employment tax” with “S.E. tax” in order to “improve” the document’s readability score (National Public Radio, 2016).

More recent measures such as Lexile (Stenner, 1996), which use word frequency in reference corpora to measure word difficulty, solve some of the problems with earlier methods. But corpus-based measures of word difficulty bring with them new problems. One is the problem of estimating the difficulty of words that do not occur in the reference corpus.

A second problem is that Lexile and other formulas provide users with a single statistic. Quantitative measures of readability use two lists of numbers: the lengths of the document’s $m$ sentences ($l_1, l_2, \ldots, l_m$) and the familiarity (or difficulty) of the document’s $n$ words ($r_1, r_2, \ldots, r_n$). Flesch-Kincaid and Lexile scores are linear combinations of two statistics, one involving sentence lengths and another involving word difficulties.3 But writers would benefit from finer-grained information than a summary score: whether their sentences could be more concise, or their vocabulary could be more commonplace, or both. It is for this reason that we disjoin the two components of popular readability formulas.

Third, today’s measures penalize the judicious, infrequent use of technical terms. Because extremely rare words naturally occur in many genres, including documents that are required to be written in plain language, penalizing a document for having a few isolated rare words—as Lexile’s averaging does—is unhelpful. For example, extremely rare words naturally occur in documents devoted to defining unfamiliar terms; countless plain language documents are devoted to this kind of explanatory, definitional work. An article written in plain language describing a coronavirus naturally uses the word “coronavirus.” Such an article should not be penalized in proportion to the negative (log) frequency of the word. Indeed, such an article’s use of the word should not be penalized at all.

Even when a document is not defining an unfamiliar term, penalizing a plain language document for an isolated rare word can be counterproductive. Technical terminology or the linguistic norm of “tecnicity” is not only an inevitable part of informational discourse, but oftentimes necessary for communicating ideas and communicating them comprehensibly (Guillory, 2004). Failing to mention that a technical term is often used to describe an item would be irresponsible since the reader may, in practice, only encounter the technical term. For example, a plain language description of how to ship goods overseas should be encouraged to mention that a list of goods for transport is called a “bill of lading,” even though the word “lading” is spectacularly rare. To the extent that penalizing documents for exhibiting technicity encourages writers to avoid technical terms, received measures of plain language can inadvertently promote less comprehensible prose. In general, there is a strong case that none of the existing quantitative measures really encourages writing that is “plain” or easier to read.

3 Methods

We gather machine-readable versions of plain language exemplars featured on the US government’s plainlanguage.gov website (maintained by PLAIN) as well as reference documents whose language is generally known (e.g., New York Times articles). For each document, we describe two empirical distributions: the distribution of sentence lengths and the distribution of word difficulties.

Note that we work with distributions and not summary statistics. Lexile, Flesch-Kincaid, and other familiar measures use averages of sentence- or word-level measurements of sentence complexity and vocabulary difficulty.

3.1 Features

Sentence lengths. We identify distinct sentences in machine-readable texts using a rule-based English language sentence tokenizer distributed with the NLTK software (Bird et al., 2009). We use the particular rule set which is distributed with version 3.5 of the software. These rules, derived from training on the WSJ portion of the Penn Treebank, have not changed since August 2013.

In order to arrive at a word count for each sen-
ence, we first tokenize the sentence using the Moses tokenizer (Koehn et al., 2007). We then remove all tokens that are not words. We define a word as a token which has characters in the following set: Unicode letters and the hyphen, with optional initial apostrophe (regular expression "‘?\p{Letter}’-+”). This definition is aligned with the Moses tokenizer, which preserves hyphenation and splits contractions. The number of tokens that remain after removing non-words is the sentence’s length.4

There are, of course, other tokenizers and other methods for identifying distinct sentences. We use established methods to facilitate others reproducing our results.

Word difficulties. We follow the existing practice of measuring the accessibility of a word by how frequently it appears in a large reference corpus. To facilitate comparison we report all frequencies as frequencies per 1 billion tokens. To transform our measure to a measure of inaccessibility we multiply by −1.

For our reference corpus, we use the English language portion of the New Crawl corpus, published in association with the ACL’s Third Conference on Machine Translation (WMT18). This corpus covers 11 years (2007-2017) and is inspired by and is a larger version of the “LM1B” language evaluation corpus (Chelba et al., 2014). After discarding duplicate sentences, we tokenize the corpus using the Moses tokenizer. This yields a corpus of 3.2 billion tokens (6.4 million types).

If a token is among the most common 100,000 types, we report its frequency per billion tokens as the measure of its accessibility. Otherwise, we estimate its frequency using regularized linear regression. Using such a model allows us to make serviceable estimates of the frequency of arbitrary tokens, including tokens which do not appear in the News Crawl corpus. Despite the corpus’s size, countless technical terms are absent, as are neologisms introduced after 2017. This model takes as input the token’s length in Unicode code points, its byte unigrams, byte bigrams, and byte trigrams. In calculating byte n-grams, we use UTF-8 encoding. The model outputs the token’s estimated log frequency per 1 billion tokens. Additional details appear in Appendix A.

For the reasons described above, in this paper we avoid using summary statistics and report empirical distributions of these two features for each analyzed document.

3.2 Documents

Plain language exemplars The US government’s website dedicated to the Plain Writing Act, www.plainlanguage.gov, offers the following documents as models of plain language documents. Given the context—a website designed to educate government officials on how to produce writing that conforms to the Plain Writing Act—we think it is appropriate to treat these documents as exemplars and not, say, marginal instances of documents conforming to the principles of the Act.

Several of the documents we use are available in the form of page images (PDFs). To reduce the labor required to transcribe text from page images, we randomly sample parts of documents. The specific sampling strategy is mentioned alongside the description of each document.

1. The 9/11 Commission Report by the National Commission on Terrorist Attacks (1,911 words). Published in 2004, the report describes events leading up to the September 11, 2001 attacks in the United States. We sample sections uniformly at random and collect paragraphs within each section.

2. Draft Grazing Manual by the Bureau of Land Management (915 words). Published in 1997, the section, “Range Improvements,” is featured on the PLAIN website. It describes regulations concerning physical improvements to lands grazed by domestic livestock or wild animals. We use the entire section.

3. National Park Service Museum Handbook, Part II by the National Park Service (1,654 words). Published in 2000, the Handbook describes how to manage National Park Service museum collections. We randomly sample sections. The handbook features technical language specific to museum operations (e.g., “archival,” “deaccessioning”).

4. Oak Ridge Reservation Annual Site Environmental Report by the Department of Energy (1,654 words). Published in 2016, the 506-page report describes the results of environmental monitoring at the Oak Ridge Reservation (ORR) in Tennessee. The ORR hosts

4For an implementation of the Moses rule-based tokenizer, we use the sacremoses Python package.
facilities associated with the maintenance of US nuclear weapons. We sample sections at random.

5. A Plain English Handbook by the Securities and Exchange Commission (1,969 words). Published in 1998, the 83-page Handbook describes “well-established techniques for writing in plain English.” The manual itself obviously uses the style and techniques it recommends, which is why we have included this document. We sample sections at random.

Reference documents

1. Voice of America Special English (2,243 words). Five articles randomly sampled from published articles on the Voice of America News in Special English website, https://learningenglish.voanews.com/. Texts written using VOA Special English, the most widely used successor of Basic English, use a vocabulary of about 1,500 words.

2. New York Times (1,783 words). A random sample of four Arts and Music section articles from The New York Times. Articles were truncated to 500 words. This sample is included as an example of writing addressed to general audience with considerable formal education.

3. Wikipedia (2,160 words). We gather paragraphs from five randomly sampled articles in the WikiText-2 corpus of “Good” and “Featured” articles (Merity et al., 2016). The articles selected are Xenon, USS Illinois, Mount Jackson, The Moth (TV episode), and Krak des Chevaliers.

4. Academic philosophy (2,025 words). We sample sections at random from Bodies that Matter (1993) by Judith Butler. We include this document as an example of non-plain writing. We considered several academic philosophy texts. Butler’s text featured distinctly longer sentences.

Although some of the reference texts are aggregations of several documents, we refer to these aggregations as “documents.”

4 Results

Figure 1 shows the distribution of sentence lengths and word difficulties for each analyzed document. All distributions exhibit positive skew. Sentence length distributions in the reference texts align well with prior expectations about document plainness. Word difficulty distributions in the reference texts are less distinctive but also roughly align with prior expectations. Plain language documents feature sentences which are shorter than those found in academic philosophy.

Sentence length and word difficulty distributions for the plain language exemplars vary with no consistent pattern. For example, the National Park Service Museum Handbook tends to use much shorter sentences than Wikipedia and the New York Times. At the same time, the Handbook’s words are not distinctly more accessible.

Two plain language exemplars, the SEC Plain English Handbook and the Oak Ridge Environmental Report, clearly differ in their use of short sentences and everyday vocabulary. 75% of sentences in the SEC Handbook use 20 words or fewer. In the Oak Ridge Report, only 52% of sentences have 20 words or fewer. The SEC Handbook uses much more accessible language. Ignoring instances of most common 500 words, 75% of words in the Handbook appear at a rate higher than 48,600 per billion tokens \( \log(48600) \approx 10.79 \). (Familiar words occurring at this rate are “easily” and “require.”) In the Oak Ridge Report, only 58% of words occur at similar rates.

5 Discussion

Our analysis indicates that writers needing to comply with the Plain Language Act can benefit from focusing on their sentences. With the exception of the National Park Service Museum Handbook, the documents that model plain writing according to PLAIN are less plain in terms of sentence length than our Wikipedia samples. Now it is possible that vocabulary simplicity causes longer sentences; this is typically true of writing in some controlled vocabulary languages, like Basic English, where sentences can run abnormally long (Igarashi, 2015). But we have found among our documents that the Voice of America Special English sample has the most commonplace words and shorter sentences, and the document with the longest sentences (the Oak Ridge Reservation Annual Site Environmental Report) also uses the rarest words. A preliminary recommendation, then, is that government agencies aiming to write plainly use shorter words—an achievable goal. A future area of research would be
to examine further the relationship between word rarity and sentence length with a larger sample of exemplary and reference documents.

In terms of vocabulary accessibility, the SEC Plain English Handbook is indeed exemplary, a valuable benchmark for other federal agencies striving to write in plain language. Its stylistic recommendations for translating abstract and obscure financial terminology form a helpful model for agencies writing about other subjects and domains (SEC, 1998a). Controlled vocabularies can also prove to be useful guides toward plain writing: in particular, Voice of America’s Special English strikes a good balance between vocabulary familiarity and sentence brevity. The style of the Oak Ridge Reservation Annual Site Environmental Report warrants reconsideration as an illustration of plain writing. We also hope to refine further our measure of word difficulty so that it is most useful for government employees.

A future line of inquiry would also consider how plainness manifests differently in different genres of informational writing. For example, do handbooks and manuals (e.g., the National Park Service Museum Handbook and the SEC Plain English Handbook) tend to exhibit briefer sentences than reports (e.g., the Oak Ridge Reservation Annual Site Environmental Report)? Our current findings are suggestive but not conclusive on this matter. But one hypothesis is that manuals and handbooks for practical purposes achieve sentence brevity more easily, whereas reports and other retrospective accounts have longer sentences due to these genres’ goal of a comprehensive account.

Theoretical humanistic writing, although much maligned for the use of jargon and other difficult words (Culler and Lamb, 2003), also merits further investigation. Our sample of philosophical academese (Butler’s Bodies that Matter) features rare terms less frequently than our Wikipedia sample and, surprisingly, at a similar rate as three plain writing exemplars (the National Park Service Museum Handbook, the Draft Grazing Manual, and the Oak Ridge Reservation Annual Site Environmental Report). According to our findings, Butler’s writing is marked not by the use of jargon but rather by long sentences.

Finally, perhaps what we are dealing with is plainer writing rather than plain writing. Plainness is not a single, fixed quality possessed by any document but rather an ideal that different documents approach in various ways and with different
resulting textual features. Also, what seems unquestionably plain for one audience may not be plain for another. And, as we have seen, several documents deemed to represent plain writing are in fact quite variable in two of the enduring stylistic indicators of plainness, sentence length and word difficulty. Writing oriented to the plainness ideal and therefore made plainer than it would have otherwise been (hence all the before and after examples found in discussions of plain language) generates varieties of plainer writing.

Acknowledgments

This work was supported by the National Science Foundation under Grant No. 1814425. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References


Clarity. 2021. About Clarity.


Julie Jennings and Jared C Nagel. 2020. Federal Workforce Statistics Sources: OPM and OMB.


PLAIN. 2021. What is plain language?


A Estimating Word Frequencies

We follow the existing practice of measuring the accessibility of a word by how frequently it appears in published writing. Although this practice sounds easy to implement, doing so is complicated by the need to make estimates of the frequency of arbitrary words, including rare words, proper nouns, and neologisms. For high frequency words, estimates derived from frequencies in large corpora of...
everyday texts (e.g., newspapers, magazines, general interest books) are serviceable. For many other words, this approach is not viable. Uncommon technical terms and proper nouns which appear in dictionaries are frequently absent from even the largest corpora. Neologisms and rare plural forms (e.g., *crowdfundings*, *virtuosas*) may not appear in dictionaries or large corpora but surely merit being assigned estimated frequencies higher than random character strings.

We solve this problem by using a simple model to estimate the frequency of uncommon words. We use regularized linear regression, also known as ridge regression, to predict a word’s frequency per billion tokens. We extract the following features from the token: length in Unicode code points, byte unigrams, byte bigrams, and byte trigrams. The model is trained to predict the token’s log frequency.

We train the model using token frequencies for all types which occur at least 50 times per billion tokens, reasoning that the News Crawl corpus contains a variety of incidental corruptions which we do not wish to model. We also exclude from the training data the most common 50,000 types, reasoning that the characters of extremely common words are not useful in predicting the frequency of rare words. We verify the model produces reasonable estimates by holding out 10% of the training data and asking the model to predict the log frequency of the held-out types.

The chief flaw with this particular approach is that it relies on a relatively homogeneous corpus of news articles. Words which tend to appear in news articles have inflated frequencies (e.g., *said*). Regularized linear regression also inflates the frequency of extraordinarily rare tokens (e.g., rare technical terms). Neither of these flaws is consequential in the present context. To study plain language we only need a general sense of how frequently a given word appears in everyday use.

Although the model is of token frequency, we only ever use frequencies of words. As described earlier in this paper, we define a word as a token which consists primarily of Unicode letters.
Word Discriminations for Vocabulary Inventory Prediction

Frankie Robertson
University of Jyväskylä
frankie@robertson.name

Abstract

The aim of vocabulary inventory prediction is to predict a learner's whole vocabulary based on a limited sample of query words. This paper approaches the problem starting from the 2-parameter Item Response Theory (IRT) model, giving each word in the vocabulary a difficulty and discrimination parameter. The discrimination parameter is evaluated on the sub-problem of question item selection, familiar from the fields of Computerised Adaptive Testing (CAT) and active learning. Next, the effect of the discrimination parameter on prediction performance is examined, both in a binary classification setting, and in an information retrieval setting. Performance is compared with baselines based on word frequency. A number of different generalisation scenarios are examined, including generalising word difficulty and discrimination using word embeddings with a predictor network and testing on out-of-dataset data.

1 Introduction

Given a small sample of words, how well can we predict whether a learner knows some out-of-sample word? This is the task of vocabulary inventory prediction. A clear motivation for the topic is to enable quicker and more precise placement testing. For example, a 40 word self-assessed word knowledge quiz used as a benchmark in this paper is quick enough that an L2 learner returning to a language learning app after a long break, in which they may have either forgotten a lot or had a lot of extra exposure to their target language, can be placed again quickly without excessive disruption.

This paper addresses the following research questions:

1. What are the empirical differences in performances between difficulty parameters produced by estimation of Item Response Theory (IRT) models and those based on word frequency in terms of their application to vocabulary inventory prediction?

2. How well can the IRT parameters of difficulty and discrimination be regressed based on word embeddings?

3. Which approaches from the field of Computerised Adaptive Testing (CAT) help to select good items to query? Does the addition of a discrimination parameter help with question selection?

4. Does the addition of a discrimination parameter help with the final prediction step?

2 Related Work

Milton (2009) refers to the common assumption when quantifying vocabulary acquisition that words are learnt in approximately descending order of frequency as the frequency assumption. It has been used in the field of reading research, for example in estimating vocabulary size, but can also provide a simple baseline for the task of vocabulary inventory prediction.

Avdiu et al. (2019) approached the problem through feature engineering, taking frequency profiles of different genres and associating learners with them according to their responses. They used a large section of the data for training, without testing a scenario in which learned data is to be generalised to new learners with less data available, as in this paper.

Item Response Theory (IRT) (Tatsuoka et al., 1968; Baker, 2001) is widely used to determine item difficulties and examinee ability in academic assessments. A key drawback of traditional IRT is that the actual content of the items is ignored. Instead, items are only understood in terms of their responses. This leaves no possibility of generalising item parameters to unseen items. Recent work
has begun to generalise difficulty scores based on representations based on items’ textual content using deep neural networks. For example Benedetto et al. (2021) first fit an IRT model on questions from a cloud technology certification exam, before training a transformer model to regress the resulting difficulty scores, allowing generalisation to new questions without a pre-testing stage.

Ehara (2019) approaches the problem of vocabulary prediction by fitting a Rasch (1960) model, equivalent to a 1-parameter logistic IRT model. The problem was modelled such that an equivalent neural network was constructed which included features based on Glove (Pennington et al., 2014) word embeddings. As with Avdiu et al. (2019), a single stage of training was performed so that the ability of the learners was learnt simultaneously with the weights of the prediction network. This network did not beat a word frequency and logistic regression baseline. In this paper, a 2-parameter logistic IRT model is fitted as an initial step, before proceeding to generalise these parameters using a word embedding based regressor.

Computerised Adaptive Testing (CAT) (Lord, 1977; Wainer, 2000) has not been widely applied to the task of vocabulary inventory estimation. A CAT system selects questions based on a examinee’s previous answers in order to converge on an accurate ability estimate faster. Related, but outside of CAT/IRT setting, Ehara et al. (2014a) builds graphs made from a combination of multiple corpora combined and apply label propagation to find a fixed set of queries to in-effect give a more accurate ability estimate than choosing at random. Restricting ourselves to the adaptive setting, the main prior art is the website http://testyourvocab.com/, which uses CAT to estimate vocabulary size based on word frequencies. To the best of the author’s knowledge, there is no prior work attempting to quantify how accurate the ability estimates obtained when applying CAT to the problem of vocabulary inventory estimation are.

3 Method

3.1 Datasets

Three datasets are used in this paper. The first, SVD12K, is due to Ehara et al. (2012) and contains 12 000 words rated on a 5-point scale by 16 learners of English, most of whom have Japanese as their native language. Following Ehara et al. (2014b), the first learner is discarded due to lower quality data. The learners in SVD12K were all students of the University of Tokyo and we speculate that it is quite possible they have all learnt English for similar purposes, i.e. academic usage, and may have even attended the same English classes.

The other two datasets are used as additional test sets, so as to see how well the techniques generalise beyond the potentially rather narrow distribution of SVD12K. Both of the two extra datasets are constructed such that they should be mainly composed of learners with Japanese as their L1, i.e. testing of generalisation beyond learner L1 is not considered here. Ehara (2018) introduce EVKD1, a dataset consisting of responses to a 100 word 4-way multiple choice test given to 100 participants, administered using a Japanese crowdsourcing platform. Respondents were asked to choose the correct definition of a word given in a context sentence. The final dataset is a section of responses to the website TestYourVocab\(^1\) limited to responses from 2018 by participants who selected their country as “Japan”. This dataset has a different selection of responses for each person.

3.2 Fitting an IRT Model

Given a matrix of responses \( r_{i,j} \) indexed by items \( i \) and respondents \( j \), an IRT model predicts latent features of the items and respondents. Respondents are assigned abilities \( \theta_i \), while in 2-parameter IRT models, items are assigned difficulties \( a_j \) and discriminations \( b_j \). Typically we predict binomial responses based on an Item Characteristic Curve

\[ a_j \sim \mathcal{N}(1.2, 0.25) \]
\[ b_j \sim \mathcal{N}(0, 1) \]
\[ \theta_i \sim \mathcal{N}(0, 1) \]

Figure 1: Plate diagram showing the Bayesian network corresponding to the 2-parameter logistic IRT model.
(ICC) like so:

\[
ICC_j(\theta) = \left(1 + e^{-a_j(\theta - b_j)}\right)^{-1}
\]

\[
P(r_{i,j}|\theta_i, a_j, b_j) = ICC_j(\theta_i)
\]

\[
Q(r_{i,j}|\theta_i, a_j, b_j) = 1 - ICC_j(\theta_i)
\]

The data of Ehara et al. (2012) is rated on a 5-point scale, suggesting a graded IRT model. A typical formulation may try and learn separate difficulty and discrimination parameters per item-level pair, significantly increasing the number of parameters to be learnt. In order to reduce the amount of data necessary to fit the IRT model, we learn only one difficulty discrimination per item and create fixed global offsets \(l_{1...s} \geq 0\) to create offset difficulties for the thresholds. We then model:

\[
P(r_{i,j} \geq k|\theta_i, a_j, b_j) = ICC_j(\theta_i - \sum_{s=1}^{4} l_s)
\]

And note that:

\[
P(r_{i,j}^* = k) = P(r_{i,j}^* \geq k) - P(r_{i,j}^* \geq k + 1)
\]

\[
P(r_{i,j}^* \geq 1) = 1
\]

\[
P(r_{i,j}^* \geq 6) = 0
\]

We estimate the Maximum A Posteriori (MAP) with Stan (Carpenter et al., 2017). The priors are illustrated alongside Figure 1. After fitting the model we revert to considering the binominal case by defining \(P(r_{i,j}) := P(r_{i,j}^* = 5)\).

### 3.3 Frequencies as a Difficulty Baseline

A simple frequency baseline for difficulty was constructed based on the word frequencies of the wordfreq (Speer et al., 2018) library. The wordfreq library incorporates frequencies from multiple corpora of different registers, ensuring balanced coverage by taking equal contributions from each register after removing outliers. Internally, wordfreq stores log frequencies on an 800 point scale. These are first negated and then standardized according to their mean and standard deviation based on the words in the SVD12K dataset so that they lie in the same range as the IRT difficulties.

To the best of the author’s knowledge, given good frequency data, this baseline has not yet been significantly surpassed on this task in the setting where there are only a small number responses available from the learner, making it effectively state-of-the-art.

### 3.4 Generalising IRT Item Parameters

In order to generalise the difficulty and discrimination parameters beyond the words present at IRT model estimation time, a Multi-Layer Perceptron (MLP) was trained as a regressor for both parameters. Words are input to the network as Numberbatch 19.08 (Speer et al., 2017) embeddings. These 300 dimensional embeddings, based on lemmas rather than word forms, are constructed by combining multiple distributional word embeddings with information from the ConceptNet lexical knowledge graph. They were chosen because most vocabulary tests are either based on lemmas or word families rather than word forms, and because they have performed well in previous studies.

The architecture shown in Figure 2 was implemented using PyTorch (Paszke et al., 2019). The GeLu activation function (Hendrycks and Gimpel, 2016) and BatchNorm (Ioffe and Szegedy, 2015) are used as non-linearities. Since full batch training is used here, the BatchNorm damping parameter, which is intended to stabilise random variations in minibatches, is not used. The Adam optimizer (Kingma and Ba, 2015) was used with a learning rate of 0.003. Training was performed for 50 iterations and the best iteration on the validation set created by 1:11 validation:train split was chosen.

### 3.5 Computerised Adaptive Testing

The aim of Computerised Adaptive Testing (CAT) (Lord, 1977; Wainer, 2000) is to estimate a learner’s ability parameter \(\theta\) as accurately as possi-
ble with as few queries as possible. Key parts of a CAT system are initialisation, next item selection, and \( \theta \) estimation. After initialisation, the system repeatedly queries a new item from the learner and re-estimates \( \theta^* \) until a termination condition. Here, we terminate after having made 40 queries, and always initialise \( \theta^* \) to be 0.

Next item selection rules are typically formulated as choosing an next item so as to maximise some measure of merit. Here we consider the maximisation of Fisher information introduced to the field of CAT by Lord (1977), and denoted as Max-Info. For the 2-parameter logistic IRT model the Fisher information is defined as:

\[
I_j(\theta) = a_j^2 ICC_{a_j,b_j}(\theta)(1 - ICC_{a_j,b_j}(\theta))
\]

An alternative next item selection rule is due to Urry (1970), and denoted as such, and simply picks questions close to the current estimate of \( \theta \). Note that this is equivalent to the max entropy heuristic in active learning, which queries the data point about which the current version of the classifier is most uncertain.

There are two approaches for estimating \( \theta^* \). The first, denoted Full-ICC, starts from a binomial IRT model introduced in Section 3.2 and incomplete response data \( U = \{u_{ij} \mid j \in J, u_{ij} \in \{0,1\}\} \). We then obtain \( \theta^* \) by maximum likelihood estimation:

\[
\mathcal{L}(\theta) = \prod_{u_{ij} \in U} P(r_{ij} \mid \theta, a_j, b_j)^{u_{ij}} \times Q(r_{ij} \mid \theta, a_j, b_j)^{(1-u_{ij})}
\]

\[
\theta^* = \arg \max_{\theta} \mathcal{L}(\theta)
\]

The second, denoted Difficulty Only, ignores the discriminations of the items, which is equivalent to setting all \( a_j = 1 \). Substituting the resulting \( ICC \) expressions into the likelihood reveals an equivalence with logistic regression. Namely, after fitting a logistic regression model on the responses \( U \), we get a model with coefficient \( m \) and intercept \( c \). We then find that \( \theta^* = -\frac{c}{m} \).

In early iterations, there may only be positive or negative responses. In this case we apply the method of Dodd (1990), which averages the previous theta estimate with either the maximum or minimum item difficulty value depending on the direction in which \( \theta^* \) would otherwise diverge.

As a non-CAT baseline, there is stratified random selection, denoted Rand. In order to guarantee a reasonable range of item difficulties are asked, strata for the words are created by ordering by frequency and splitting into 5 equal sized strata. The random selection procedure then chooses 40 items randomly, taking equally from each stratum.

The catsim Python library (De Rizzo Meneghetti and Aquino Junior, 2017) is used for the implementations of all CAT techniques.

### 3.6 Evaluation

The vocabulary inventory prediction task can be viewed as a binary classification problem. The Receiver Operator Characteristic (ROC) curve plots the recall of the positive class against the recall of the negative class by varying the classifier threshold. Statistics based on ROC curve, such as Area Under ROC (AUROC) enjoy the key advantage of threshold invariance. On the other hand, we typically do have to pick some threshold and for this reason, a metric based on a default threshold of 0.5 is given: Matthews Correlation Coefficient (MCC). The second angle on the problem that is of known and unknown word retrieval. In this case Average Precision (AP) acts as a threshold invariant measure of retrieval performance. We consider AP+ and AP- for measuring retrieval performance from the two classes of known and unknown respectively.

AUROC does not change significantly based on exact ability estimate of the learner due to its lack of a fixed threshold. Here, we use it only to explore different ways the difficulty parameter can be obtained and the effect of including the discrimination parameter. Being based on a fixed threshold, MCC is highly sensitive to the actual ability estimate, and so it gives a more realistic picture of performance practically. The metrics AP+ and AP- are used to measure an upper bound on the performance on the retrieval tasks.

Intuitively, we can see low values of discrimination as reflecting a degree of uncertainty about a word’s true difficulty. The information retrieval perspective is particularly relevant here since the presence of the discrimination parameter means that, for example in unknown word retrieval, words that are highly discriminating but less difficult could be returned earlier than words with low discrimination that are more difficult, potentially improving performance.

### 4 Experiments

We first evaluate how well the item/word parameters from the IRT model can be regressed with
Table 1: Table containing the both the raw Mean Absolute Error (MAE) and the MAE normalised by the true standard deviation of difficulties and discriminations as predicted in two word generalisation scenarios.

<table>
<thead>
<tr>
<th></th>
<th>Gen</th>
<th>Param</th>
<th>MAE</th>
<th>Norm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Words</td>
<td>Diff</td>
<td>0.595</td>
<td>0.495</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Discrim</td>
<td>0.148</td>
<td>1.365</td>
</tr>
<tr>
<td></td>
<td>Both</td>
<td>Diff</td>
<td>0.608</td>
<td>0.506</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Discrim</td>
<td>0.147</td>
<td>1.356</td>
</tr>
</tbody>
</table>

the chosen architecture. Next we move on to consider how well various CAT approaches can estimate learners’ abilities. Finally, the results for the final task of vocabulary inventory prediction are presented, first cross validating on the SVD12K dataset and then training on the whole SVD12K dataset and testing on the extra datasets.

Four generalisation scenarios are considered across experiments:

**Gen-None** No generalisation; The IRT model is fitted on the same data as the test data.

**Gen-Word** Generalising only to new words; 3-fold cross validation is performed on words, with the IRT model being fitted on $\frac{2}{3}$ training words, before fitting the MLP on the results to predict the out-of-vocabulary $\frac{1}{3}$ of words.

**Gen-Respondent** Generalising only to new learners; 3-fold cross validation is performed on participants, with the IRT model being fitted on $\frac{2}{3}$ participants, from which the item parameters are used as-is on the out-of-sample $\frac{1}{3}$ of participants.

**Gen-Both** Generalising to new words and learners; 9-fold cross validation is performed, consisting of the product of 3-fold cross validation on participants with 3-fold cross validation on words.

### 4.1 Predicting Item Parameters

Table 1 gives the results evaluating the performance of the IRT parameter regressor. When looking at the results normalised by true standard deviation, it is clear that the parameter of discrimination is more difficult to predict. The lower error in predicting difficulties in the Gen-Words scenario suggests that the more accurate IRT predictions made with more data do indeed provide an easier target for the network to fit. However, the actual errors are quite close, and the generalisation scenarios tend to give similar results, so for this reason Gen-Words is not considered further in the later results.

### 4.2 $\theta$-estimation

We now turn to the matter of how well $\theta$ is estimated using different approaches. The results are shown in Table 2.

For both next item selection methods and $\theta$-estimation methods, including the discrimination parameter seemed to decrease performance. Noteworthy is that the best overall score is obtained by difficulty-based CAT for the Gen-Both and Gen-Resp, with this setting in the Gen-Both scenario outperforming the others, showing that the regressed word difficulties perform well for this task. For the Gen-None scenario, including the full ICC when estimating $\theta$ appeared to help. It may be that having non-regressed discrimination values based on responses from more respondents helped in this case.

However, since discriminations appear to not be generally useful for finding $\theta$ in any generalisation scenario, they are not used further in the next section and the Urry (1970) next item rule is used together with the difficulty only $\theta$ estimator.

### 4.3 Vocabulary Inventory Prediction

We now evaluate the final task of vocabulary inventory prediction. Table 3 shows the results on this task using the metrics introduced in Section 3.6. The experiments compare the use of dif-

<table>
<thead>
<tr>
<th></th>
<th>Gen</th>
<th>Estimator</th>
<th>Next Item</th>
<th>MAE</th>
<th>Norm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Both</td>
<td>Full ICC</td>
<td>Rand</td>
<td>1.204</td>
<td>1.136</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Urry</td>
<td>1.117</td>
<td>1.053</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Max-Info</td>
<td>1.110</td>
<td>1.047</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Difficulty only</td>
<td>Rand</td>
<td>1.087</td>
<td>1.025</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Urry</td>
<td>1.037</td>
<td>0.978</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Max-Info</td>
<td>1.114</td>
<td>1.051</td>
</tr>
<tr>
<td></td>
<td>Resp.</td>
<td>Full ICC</td>
<td>Rand</td>
<td>1.334</td>
<td>1.258</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Urry</td>
<td>1.150</td>
<td>1.084</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Max-Info</td>
<td>1.199</td>
<td>1.131</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Difficulty only</td>
<td>Rand</td>
<td>1.280</td>
<td>1.207</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Urry</td>
<td>1.068</td>
<td>1.007</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Max-Info</td>
<td>1.211</td>
<td>1.142</td>
</tr>
<tr>
<td></td>
<td>None</td>
<td>Full ICC</td>
<td>Rand</td>
<td>1.372</td>
<td>1.294</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Urry</td>
<td>1.105</td>
<td>1.042</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Max-Info</td>
<td>1.395</td>
<td>1.316</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Difficulty only</td>
<td>Rand</td>
<td>1.249</td>
<td>1.178</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Urry</td>
<td>1.233</td>
<td>1.163</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Max-Info</td>
<td>1.338</td>
<td>1.262</td>
</tr>
</tbody>
</table>
Table 3: Table showing results on the SVD12K dataset in different generalisation settings given different choices of source difficulty parameter and whether to include the discrimination parameter in predictions.

Table 4: Table showing results on the EVKD1 dataset of different choices of source difficulty parameter and whether to include the discrimination parameter in predictions.

4.4 Generalising Vocabulary Inventory Prediction

We now turn to a scenario in which all the data from the SVD12K dataset is used for training, equivalent to the Gen-None scenario, but the resulting item parameters are tested on external datasets. We test on the EVKD1 data set and TestYourVocab dataset introduced in Section 3.1. The results are given in Tables 4 & 5.

Since the EVKD1 set is a 4-way multiple choice test, we account for correct answers by guessing by using an item response curve with a guessing probability of $0.25$, similar to the 3-parameter logistic IRT model:

$$ICC_{a_j,b_j}(\theta) = 0.25 + \frac{0.75}{1 + e^{-a_j(\theta_j - b_j)}}$$

Since there is a limited number of training words available in these datasets, in these experiments, no CAT is used, and instead the difficulty parameter is estimated based on 40 words taken at regular intervals from the frequency ranked list. There are three generalisation scenarios: $Freq$, where only frequency data is used; $Pred$, where only predictions from the generalisation model are used; and $Mix$, where item parameters are used directly from the IRT model fitted on SVD12K where possible, falling back to predictions when items available in SVD12K. Other variations are as in Section 4.3. For both datasets, frequency based difficulties outperform difficulties estimated from SVD12K, suggesting these do not generalise well to other datasets. The inclusion of the discrimination parameter appears to have a consistent small negative effect across all these experiments.
5 Discussion

We now summarise and discuss some of the main results of the experiments. Firstly, the discrimination parameter does not appear to help with query item selection, however it remains somewhat inconclusive whether it can help with estimating the learner ability \( \theta \) since this was the best configuration in the Gen-None case. It may be that with sufficiently high quality estimates of the discrimination values, using this for \( \theta \)-estimation would help more. The approach which appeared best overall in this case however, and which was used for later experiments on the SVD12K dataset ignored the discrimination parameter altogether for both steps of the CAT stage.

The difficulty parameter generalises reasonably well, while the discrimination parameter generalises quite poorly when regressed using a MLP based on Numberbatch representations of the word items. Since item difficulty here is closely related to frequency, it seems quite possible that a lot of the generalisation is happening based on frequency information encoded in the word embeddings. When considering how well both parameters generalised, we should note that only one type of word embedding and regressor was tried, and others may generalise this parameter better.

The regressed difficulties perform better than the frequency data on in-dataset data, while performing worse on out-of-dataset data. Given all datasets contained mostly Japanese learners of English, this suggests that both the IRT parameter and the MLP generalising may have over fitted on narrow attributes of the particular cohort of University of Tokyo students making up SVD12K. Conversely we see that that high quality, balanced word frequency data generalises rather well.

Usage of the discrimination parameter for vocabulary inventory prediction was largely inconclusive, with some evidence against it. In many cases, it appeared to decrease performance on metrics such as AUROC, however some tasks showed a promising but insignificant boost in AP-

It is unclear exactly why the discrimination parameter failed to provide significant improvements in either next-item selection, \( \theta \)-estimation or vocabulary inventory prediction. It is possible that the amount of response data was not sufficient either in terms of the number of respondents, or in terms of representing a diverse range of abilities, to obtain accurate word discrimination estimates. Apart from simply finding and integrating more vocabulary knowledge data, one direction for future work is trying to find corpus derived measures which correlate with word discrimination, analogously to the negative correlation between word frequency and word difficulty. This would also effectively address the failure to generalise the word discriminations parameter to out of vocabulary words.

We hope the methods of evaluating the different sub-tasks of the vocabulary inventory prediction task in the settings demonstrated here can help establish practices for evaluating this task more thoroughly. We also hope that the framing given here inspires others to tackle the problem in the challenging, but more broadly applicable setting of vocabulary inventory prediction having a small, limited number of queries.

The code to replicate all experiments is made available at https://github.com/frankier/vocabirt.

References


Yo Ehara. 2019. Neural rasch model: How do word embeddings adjust word difficulty? In PA CLING.


Jeffrey Pennington, R. Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In EMNLP.


Abstract

Lexical simplification (LS) aims at replacing words considered complex in a sentence by simpler equivalents. In this paper, we present the first automatic LS service for French, FrenLyS, which offers different techniques to generate, select and rank substitutes. The paper describes the different methods proposed by our tool, which includes both classical approaches (e.g. generation of candidates from lexical resources, frequency filter, etc.) and more innovative approaches such as the exploitation of CamemBERT, a model for French based on the RoBERTa architecture. To evaluate the different methods, a new evaluation dataset for French is introduced.

1 Introduction

It is widely acknowledged that reading difficulties, either due to insufficient education or to mental deficiencies for example, can hinder access to information, which is likely to result in a loss of autonomy and freedom (Mutabazi and Wallenhorst, 2020). Faced with this challenge, researchers imagined applying natural language processing (NLP) to automatically transform sentences in a text in order to make it more readable, thus facilitating access to information. This is the objective pursued in the field of Automatic Text Simplification (ATS), in which the main goal is to preserve grammaticality and meaning while carrying out effective transformations to make the text simpler.

ATS is generally investigated at the sentence level (Seretan, 2012; Brouwers et al., 2014) using rule-based systems. In parallel, lexical simplification was also investigated, based on lexicons or resources (Billami et al., 2018; Carbone, 2018; Hmida et al., 2018). Due to the lack of training data, machine translation approaches - which are standard for English - were applied to French (Rauf et al., 2020) only very recently, with mixed results. As a result, the situation of ATS for French is clearly lagging behind that of English. The only simplification package freely available for the research community has been published recently (Wilkens and Todirascu, 2020) and it remains preliminary and focused exclusively on syntax. AMesure, a web platform designed to help writers of administrative texts to write in plain language (François et al., 2020) is more encompassing. However, it is limited to detecting complex phenomena and suggesting simplifications.

In this paper, we aim to fill the gap in lexical simplification (LS) tools and resources for French by developing a tool in which several standard approaches of LS are available and by building a reference dataset to evaluate our results. This library, called FrenLyS for French Library for Simplification, follows the LS process first described...
by Shardlow (2014) as a sequence of four steps: identifying complex terms, generating candidates for substitution, selecting the best candidates, and ranking them according to their degree of readability. Our work draws from similar packages in other languages, such as LEXenstein (Paetzold and Specia, 2015) for English, the EASIER tool (Alarcon et al., 2019) and LexSis (Bott et al., 2012) for Spanish or the work by Qiang et al. (2021).

The article is structured as follows. Section 2 presents the state of the art of lexical simplification. In Section 3, we describe the different approaches we implemented for each of the LS steps. Section 4 describes the methodology used to evaluate our approaches, which includes a new reference dataset for French. In section 5 we report and discuss the performance of each of these approaches on our evaluation dataset.

2 Related Work

The task of lexical simplification was first investigated by Carroll et al. (1998) who exploited a rather simple solution: they obtained candidates for substitution using WordNet synonyms (Miller, 1995) and ranked them according to their frequency. As a result of this work, researchers tried to improve different aspects of this process, either by collecting synonyms (De Belder and Moens, 2010), or by ranking the candidates (Biran et al., 2011a), etc. In his survey of the field, Shardlow (2014) provided a clear view of the different challenges within LS, identifying four steps in which recent work can be classified.

Complex Word Identification The first step in lexical simplification is the complex word identification (CWI). This step has been the object of several shared tasks (Paetzold and Specia, 2016c; Yimam et al., 2018) and aims at identifying in a text the words or expressions likely to be problematic for a target audience of readers and on which the LS system should be applied. As Gooding and Kochmar (2019) pointed out, early works on the complex word identification operated by simplifying all words (Thomas and Anderson, 2012; Bott et al., 2012) or were based on a threshold \( t \) over a given metric of simplicity (e.g. word frequency) that separates simple from complex words (Biran et al., 2011b). Another approach consists in finding complex words with the help of a lexicon: if the word appears in the resource it is considered as complex, otherwise as simple. This method has been mostly used for lexical simplification of medical texts (Chen et al., 2016; Deléger and Zweigenbaum, 2009). Other more recent attempts either used machine learning to classify words as either complex or simple based on some features such as word length, word frequency, number of senses, etc. (Shardlow, 2013; Alarcon et al., 2019).

Substitution Generation Once complex terms have been identified, the next step is to produce candidates that can replace for the target complex word. This step, called substitution generation (SG), is most often carried out querying linguistic lexical resources, as evidenced by the work of Carroll et al. (1998), Bott et al. (2012), or Hmida et al. (2018). They generate synonyms by querying lexical databases such as WordNet or synonym resources such as ReSyf (Billami et al., 2018). As it is not always easy to find lexical databases and as those might have limited coverage, Horn et al. (2014a) proposed to use parallel corpora – Wikipedia and Simple Wikipedia – to automatically extract lexical simplification rules. Deléger and Zweigenbaum (2009) resorted to paraphrases to replace target complex words, a strategy that is more relevant for specialized languages. A currently popular approach was first suggested by Glavaš and Štajner (2015). It consists in obtaining synonyms in an unsupervised way relying on semantic representations such as embeddings. The complex word to be substituted is projected in the semantic space in order to generate the N closest semantic neighbors. More recently, Qiang et al. (2019) used BERT in a similar fashion.

Substitution Selection In order to obtain semantically correct sentences, each candidate has to go through a disambiguation step. The substitution selection (SS) step aims to decide which of the candidates collected at generation step best fits the context of the sentence to be simplified. De Belder and Moens (2012) proposed to carry out the task of disambiguation using a Latent Words Language Model (LWLM): they use Bayesian networks to represent words and their contextual meaning. Other studies took advantage of word sense disambiguation systems to explicitly label the senses of both the target and the candidates, in order to select a candidate having the same sense as the target (Thomas and Anderson, 2012; Nunes et al., 2013). A third line of research leveraged semantic models to compare the semantic similari-
ity of each candidate with the sentence to simplify. Bott et al. (2012) exploited a vector space model, whereas Paetzold and Specia (2015) rather used a word embedding model. It is also possible to perform this step in a simpler way, by removing all candidates who do not share the same part of speech as the word to be replaced (Paetzold and Specia, 2013).

### Substitution Ranking

After having identified the complex words, generated synonyms, and selected the most coherent ones, the final step of LS consists in ranking the remaining candidates according to their reading ease. The first LS systems generally resorted on frequency (Carroll et al., 1998; De Belder and Moens, 2010; Specia et al., 2012) where it is considered that more frequent words are easier to understand. Specia et al. (2012) showed that this simple rule actually represents a very strong baseline, as it outperformed 9 out of the 11 ranking systems engaged in this task of SemEval 2012 (Specia et al., 2012). Other studies proposed simplicity metrics that can be combined word characteristics: Biran et al. (2011a) and Bott et al. (2012) combine frequency with word length. Finally, it is also possible to use statistical ranking algorithms (Horn et al., 2014a; François et al., 2016) that we can combine with neural networks (Paetzold and Specia, 2017a).

### Available datasets

In parallel to the design of new LS methods, the development of reference datasets to train and evaluate those methods is key. Several datasets for lexical simplification are available for English, such as SemEval 2012 (Specia et al., 2012), LSeval (De Belder and Moens, 2012), LexMTurk (Horn et al., 2014b), NNSeval (Paetzold and Specia, 2016b), and BenchLS (Paetzold and Specia, 2016d). Other languages are not so well resourced: there are only 2 datasets for Japanese – SNOW E4 (Kajiwara and Yamamoto, 2015) and BCCWJ (Kodaira et al., 2016) –, but, to our knowledge, none for French. In French, the only available resource for text simplification is the ALECTOR corpus (Gala et al., 2020). It consists in 79 parallel texts with information about complex words, but there are no validated simpler synonyms, which are required to assess LS approaches.

### 3 Proposed Approach

Our system is the first to offer several methods for generating candidates for substitution, selecting them based on semantic similarity with the target word and ranking them by difficulty in French. Most of these methods have been previously applied to English, and we adapted them to the case of French. A few are new. All of them are described hereafter.

It should be noted that FrenLyS does not implement any complex word identification algorithm. We believe this is a very complex task, which should be addressed as a whole and actually is (Yimam et al., 2018), especially because CWI requires to take into account the readers’ characteristics. Methods based on lexical characteristics or word lists overlook the reader’s characteristics and Lee and Yeung (2019) have rightly stressed that current approaches offer the same substitutions regardless of users. This tool is therefore based on the prerequisite that complex words already have been identified. For the sake of the evaluation of our tool, we relied on a manual annotation of complex words in our test set (see Section 4).

### 3.1 Substitution Generation

The task of substitution generation aims to generate candidate synonyms to replace complex words. To carry out this step, FrenLyS proposes three methods: synonyms are directly obtained from a resource of synonyms (see ReSyf generator), or are generated by embeddings, either produced by FastText (see FastText generator) or by the BERT architecture (see CamemBERT generator). We also propose a 4th approach (see CamemBERT union) that combines CamemBERT with Resyf or with FastText to take advantage of the contextual information that the model can provide.

#### ReSyf generator

This module generates candidates from the graduated and disambiguated synonyms resource ReSyF (Billami et al., 2018). It is built from the semantic network JeuxDeMots and contains 57,589 entries that are connected to 148,648 synonyms (both in their lemma form). Its main asset is that synonyms corresponding to different meanings of a word have been manually and automatically clustered into different synsets. Another interesting feature of ReSyF is that the synonyms gathered in the same synset are ranked according to reading ease. Based on those characteristics, our method simply consults ReSyF using the lemma of the complex word and returns the lemmas of the top three simplest synonyms in each corresponding ‘synset’. At this step, we do not try
to disambiguate the meaning of the word to substitute, as this is the role of the substitution selection step.

**FastText generator**  *FastText* (Bojanowski et al., 2017) is a library for efficient learning of word representations. Its advantage for our task is that it proposes character n-gram embeddings: we can thus obtain a vector representation even for a word that does not exist in the training corpus. Thanks to this technique, we return, for any given complex word, its k-nearest semantic neighbors (the inflected forms) based on cosine similarity.

**CamemBERT generator**  In the same way that Qiang et al. (2019) generated synonyms with *BERT*, we rely on a pre-trained version of *CamemBERT* (Martin et al., 2020), based on the RoBERTa architecture, on the French subcorpus of the multilingual corpus OSCAR. Our method employs the masked language model (MLM) that masks some percentage of input tokens and predicts the masked words from their right and left contexts (Qiang et al., 2019). The idea is to mask the complex word and use the top predicted words (inflected forms) as candidate for substitution. As *FastText*, *CamemBERT* proposes a solution to deal with out-of-vocabulary tokens through their decomposition in wordpieces.

**CamemBERT union**  This method is based on the 2 following observations: ReSyf and *FastText* generators only care about the complex word and not its context while *CamemBERT* only cares about the context but does not know the complex word. The solution is to combine the advantages of both approaches by computing the *CamemBERT*-score for each substitute generated by ReSyf or *FastText*. For this purpose, we predicted the n best candidates with the *CamemBERT* generator and the k best candidates for the other method. Then we retain only the words that are in the intersection of the two generators and sort them by their new score.

### 3.2 Substitution Selection

This step takes the list of candidate synonyms and selects only those that are acceptable in the context of the complex word to replace.

We have decided to implement two of the four approaches covered in section 2, as they rely on very different strategies: either by eliminating candidates that do not have the same part of speech (see POS selector), or by leveraging language models such as *FastText* to verify the semantic compatibility between the candidate and the context (*FastTextWord selector, FastTextSentence selector*).

**POS selector**  Following Paetzold and Specia (2013), we decided to include a function within our generation methods that checks whether the generated candidates and the word to be replaced share the same parts of the speech. To do this, we used the possible tags for this word as given in the Delaf dictionary. If the intersection of POS-tags for the 2 words (complex word and candidate) is empty, the candidate is rejected.

**FastTextWord selector**  For each synonym, this selector first retrieves the *FastText* embeddings of the complex word and this synonym and compute the cosine similarity between both vectors. The more similar the meanings of both words are, the closer their vectors are. We therefore select candidates for which the cosine similarity with the complex word is greater than the heuristic threshold of 0.5.

**FastTextSentence selector**  Instead of directly comparing the vectors of the complex words and their synonyms, as in the previous approach, we use the context of the complex word for vectorization: we compute the cosine distance between the vectors of the synonyms and the vector of the complex sentence. We select the candidates with a similarity score greater than the heuristic threshold of 0.35.

### 3.3 Substitution Ranking

Finally, the last part of our system classifies the synonyms according to their degree of reading ease.

For this step, we referred to common ranking methods in the literature (cf. section 2) as we relied on frequency (see Lexique3 ranker) and in a slightly more original way, we provide a method that ranks words according to the number of meanings they have and frequency (see FreNetic ranker). We also propose a method that combines various linguistic prescriptors through an SVMRank algorithm (Herbrich et al., 2000) (see SVM ranker).

**Lexique3 ranker**  For this method, we use the commonly acknowledged fact that the more frequent a word is, the simpler it is. To obtain the

---

1The Delaf (Courtois, 2004) contains about 792,260 entries (inflected forms). For each entry, the dictionary provides the following information: lemma, pos and inflectional information (e.g. dictionaries,dictionary.N).
frequency of the candidates for substitution, we use the French lexical database Lexique3 (New, 2006) which provides, for 140,000 words of the French language, their frequencies of occurrences in literary texts and movie subtitles. We have chosen to use the frequencies estimated on the corpus of film subtitles because it contains a more up-to-date vocabulary.

**FreNetic ranker** In the same way as Elhadad (2006), this ranker exploits polysemy as a measure of familiarity and therefore of difficulty. Words from the general lexicon are more polysemous while technical terms tend to be monosemic. To collect the number of senses, we relied on FreNet², a python API for WOLF (Sagot and Fiser, 2008), a free French Wordnet. Synonyms are therefore ranked according to their number of meanings (more is easier). When several words get the same number of senses, we decide to further rank them based on their frequencies.

**SVM ranker** We also propose to perform the ranking task using a SVMRank algorithm described in François et al. (2016). It is able to rank any set of words using 21 word characteristics such as word frequency, presence of the word in a list of simple words, number of phonemes, number of letters, number of senses, number of orthographical neighbors, etc. To train it, we used the Manulex vocabulary list (Lété et al., 2004) that includes 19,038 lemmas annotated with their level of complexity. Based on that information, we prepared training pairs of two words, one of which is known to be more complex than the other, which were fed to the SVMRank algorithm. In their paper, François et al. (2016) report an accuracy of 77% with 10-fold cross-validation and a mean reciprocal rank of 0.84, obtained on a reference dataset of 40 synsets including a total of 150 synonyms that were ranked by 40 human annotators.

4 Evaluation Process

It is usual to create evaluation corpora with the help of human annotators but this requires time and lots of annotators, which may also overlook some valid synonyms. Therefore, we opted for a hybrid approach, i.e. we chose to use our synonym generation methods to propose an exhaustive list of synonyms and then we called upon annotators to select them in context and rank them according to their difficulty. The advantage of this approach is that it combines several methods from very different generations, including a synonym dictionary that was created from propositions submitted by humans (Lafourcade, 2007). In this way, we still collect data made by humans but *a priori*. We explain the corpus creation process in the next section before discussing the evaluation measures we used.

4.1 A Tailor-Made Evaluation Corpus

We decided to fill the lack of resources evaluation in French LS by proposing a dataset of annotated sentences, collected from two sources. The first set of sentences was sampled from the French reference dataset ALECTOR (Gala et al., 2020): it includes sentences with complex words and candidates for substitution³. Complex words have not been directly annotated in these sentences. However, they were read by various profiles of readers through an interface and reading times have been collected. Based on this information, we have manually identified seemingly complex words. Once the two sets of sentences were collected and the complex words were identified, we had to generate substitutes, and manually select them and classify them, as was described in the following subsections.

**Generate and select candidates** For each complex word, we produced synonyms that we annotated, using all our generation methods. The relevance of these synonyms in the context of the original sentence was then assessed by 3 expert linguists. They had to assign a score of 1 if the word is considered correct, otherwise 0. In this process, we applied the following guidelines : a word is considered synonymous as long as its replacement does not change the meaning of the sentence. To obtain a wide range of synonyms, we decided to accept hyperonyms and hyponyms – provided they fit the context – and to accept synonyms even if their register was different from that of the original word. This task is very complicated since there is no perfect synonymity and the validity of a candidate can therefore be perceived differently from

--²https://github.com/hardik-vala/FreNetic
one annotator to another one. However, thanks to the annotation guidelines and a discussion session between the annotators to discuss the criteria, the inter-rater agreement (Fleiss’ K) between the three annotators, computed on a sample of 500 candidates, is 0.638, which corresponds to a substantial agreement (Artstein and Poesio, 2008).

**Ranking of substitutes** Finally, for this last annotation, we resorted to on 20 non expert annotators aged from 20 to 57 years, whose native language is French. They had to rank the synonyms validated at the previous step by reading ease. To that end, we used the online LimeSurvey tool⁴ to deliver 2 different questionnaires of 25 items. The survey is presented as follows: each question includes a target sentence and the complex word in bold, as well as a list of synonyms in the left column. The task of the participants is to drag all the synonyms into the right-hand column to rank them, the top word being considered the most difficult.

Once all annotations have been collected, we proceed to average all annotators: in the same way as Specia et al. (2012), we assign each substitution a score based on the average of the scores assigned to it.

### 4.2 Evaluation Metrics

To compare our different methods of generation and selection, we used the following metrics as described in Paetzold and Specia (2016a): potential, precision, recall and F1. For the evaluation of ranking methods, we also employ the metrics rank-at-i and recall-at-i as mentioned in Paetzold and Specia (2016a).

### 5 Results

This section presents the results for each step of simplification and compares the different methods proposed in FrenLys. In view of the absence of comparable work for French, we put our results into perspective with those of Paetzold and Specia (2017c) for English and Qiang et al. (2021) for Chinese.

#### 5.1 Substitution Generation

As we can see in Table 1, FastText generator, a method based on non contextual embeddings, slightly outperforms Resyf generator, based on a dictionary (F1 is 0.25 vs. 0.23). Resyf has a higher potential and recall, but suffers from a lack in precision, which is due to the fact that no sense disambiguation is carried out in synonym selection. In contrast, FastText reject candidates that do not correspond to the most frequent meaning of a form (as FastText computes only one vector per form, the most frequent sense has the largest influence on it). CamemBERT is clearly the less efficient technique. It is not a complete surprise, as it can generate words that fits the context, but are not valid synonyms of the complex word.

<table>
<thead>
<tr>
<th>FrenLys</th>
<th>Pot.</th>
<th>Prec.</th>
<th>Rec.</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resyf</td>
<td>0.63</td>
<td>0.20</td>
<td>0.28</td>
<td>0.23</td>
</tr>
<tr>
<td>FastText</td>
<td>0.59</td>
<td>0.25</td>
<td>0.27</td>
<td>0.25</td>
</tr>
<tr>
<td>CamemBERT</td>
<td>0.45</td>
<td>0.13</td>
<td>0.18</td>
<td>0.15</td>
</tr>
<tr>
<td>CamemBERT + Resyf</td>
<td>0.55</td>
<td>0.29</td>
<td>0.23</td>
<td>0.26</td>
</tr>
<tr>
<td>CamemBERT + FastText</td>
<td>0.43</td>
<td>0.18</td>
<td>0.16</td>
<td>0.17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Paetzold-NE</td>
<td>0.88</td>
<td>0.31</td>
<td>0.14</td>
<td>0.19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Qiang et al. (2021)</th>
<th>Pot.</th>
<th>Prec.</th>
<th>Rec.</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid</td>
<td>0.90</td>
<td>0.43</td>
<td>0.26</td>
<td>0.33</td>
</tr>
</tbody>
</table>

Table 1: SG FrenLys results

We therefore tried to combine the advantages of CamemBERT (suitability to the context) with those of Resyf and FastText (better synonym generation). Considering the union between CamemBERT and FastText seems to hurt the performance (0.17 in F1 instead of 0.25), whereas combining CamemBERT and ReSyF produces our best results (0.26 in F1). It seems that ResyF selects valid, but not necessarily context-appropriate synonyms, which are filtered by BERT based on the context. Although not directly comparable, the F1 of our best method is in line with those of Paetzold and Specia (2017c) and Qiang et al. (2021). At the potential level, the difference observed could be explained by a variation in the number of synonyms produced by the generators: potential is correlated with the number of generated synonyms.

#### 5.2 Substitution Selection

We apply each of our selection methods on the union of all generated synonyms. The results obtained are presented in Table 2.

Results clearly reveal the importance of selecting synonyms that share the same POS as the word to substitute. This allows our system to reach a F1 of 0.31 for the generation of synonyms. It is however surprising that the POS approach outperforms both CamemBERT and FastText.

---

⁴https://www.limesurvey.org/fr/
<table>
<thead>
<tr>
<th>FrenLys (all generators)</th>
<th>Pot.</th>
<th>Prec.</th>
<th>Rec.</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS</td>
<td>0.66</td>
<td>0.29</td>
<td>0.34</td>
<td>0.31</td>
</tr>
<tr>
<td>FastText Sentence</td>
<td>0.59</td>
<td>0.27</td>
<td>0.26</td>
<td>0.27</td>
</tr>
<tr>
<td>FastText Word</td>
<td>0.58</td>
<td>0.27</td>
<td>0.26</td>
<td>0.27</td>
</tr>
<tr>
<td>Paetzold (2017)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pot.</td>
<td>0.97</td>
<td>0.23</td>
<td>0.26</td>
<td>0.24</td>
</tr>
<tr>
<td>Paetzold-BR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Benchmarking results for SS with substitutions generated by all generators

Once more, our results appears to be comparable with those of Paetzold and Specia (2017c), as our F1 is clearly higher, but we were not able to obtain such a high potential. It is interesting to notice that our selectors mostly improve precision.

5.3 Substitution Ranking

Finally, we tested our different ranking methods on the part of corpus that has been also annotated for the reading ease of synonyms. Results are displayed in Table 3.

<table>
<thead>
<tr>
<th>FrenLys</th>
<th>TR-1</th>
<th>Rec-2</th>
<th>Rec-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexique3</td>
<td>0.42</td>
<td>0.62</td>
<td>0.73</td>
</tr>
<tr>
<td>Frenetic</td>
<td>0.44</td>
<td>0.65</td>
<td>0.71</td>
</tr>
<tr>
<td>SVM</td>
<td>0.50</td>
<td>0.71</td>
<td>0.79</td>
</tr>
<tr>
<td>Paetzold (2017)</td>
<td></td>
<td>TR-1</td>
<td></td>
</tr>
<tr>
<td>Neural</td>
<td>0.66</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Benchmarking results for SR

Ranking candidates based on frequency remains a strong baseline, as the Lexique3 ranker has a TRank-at-1 of 0.42 and a Recall-at-3 of 0.73. This means that, in 42% of our test sentences, using only the frequency allows to correctly predict the synonym defined to as the simplest by human judges (gold). Using the number of senses per words in addition to frequency does not bring much improvement, only 2% for TRank-at-1. In contrast, a much more sophisticated ranker using 21 word features, clearly improves performance, as it is able to select the easier synonyms in 50% of the cases. In this step, our results remain lower than those of Paetzold and Specia (2017c) in terms of TRank-at-1.

6 Conclusion

To conclude, we described the first tool for French lexical simplification that carries out three of the four classic LS steps. Our tool, FrenLyS, will be made available to the scientific community via a freely accessible web service\(^5\).

FrenLyS includes five synonym generators, based on the two principal approaches in the field: using a resource and querying embeddings. Whereas Hmida et al. (2018) had concluded that using ReSyF as a resource was able to outperform the approach of Glavaš and Štajner (2015), we found that relying on FastText was more efficient. However, our best method combines a synonym database with CamemBERT as a way to filter inappropriate synonyms in context. These two sources bring information about the complex word semantic (ReSyf) and its context (CamemBERT), which comes close to the twofold strategy of Qiang et al. (2019). They indeed generate synonyms based on one sentence in which the complex is masked (contextual information) and the same sentence in the complex word is present in order to keep the semantic information conveyed by the complex word.

FrenLyS offers three of them and the results showed that using a simple POS filter is sufficient to improve the F1 of our generators. Ranking synonyms can be done through three techniques, the best of which integrates 21 word characteristics into a SVM ranker. The results obtained for ranking seem lower than those of Paetzold and Specia (2017c). This could be due to variations in the test data, but maybe also to the use of a neural classifier. We plan to improve our ranking algorithm using a neural ranker in the future to investigate this issue.

Finally, in addition to the implementation of the first complete LS tool for French, this paper also proposes the first evaluation dataset for French LS. This dataset will be distributed through the same web site as the API\(^6\). We hope that the availability of both resources could help boosting current LS research in French, which lacks behind similar research for other European languages.

Acknowledgments

This project was partially funded by the "Direction de la language française” from the Federation Wallonia-Brussels (AMesure project). We also want to thank Dr. Rémi Cardon for his in-depth reading and comments.

\(^5\)https://cental.uclouvain.be/frenlysAPI/
\(^6\)https://cental.uclouvain.be/frenlysAPI/
References


Sian Gooding and Ekaterina Kochmar. 2019. Complex word identification as a sequence labelling task. In ACL.


Jipeng Qiang, Yun Li, Yi Zhu, Yunhao Yuan, and Xindong Wu. 2019. A simple bert-based approach for lexical simplification.
Spelling Correction for Russian: A Comparative Study of Datasets and Methods

Alla Rozovskaya
Department of Computer Science
Queens College, CUNY
arozovskaya@qc.cuny.edu

Abstract
We develop a minimally-supervised model for spelling correction and evaluate its performance on three datasets annotated for spelling errors in Russian. The first corpus is a dataset of Russian social media data that was recently used in a shared task on Russian spelling correction. The other two corpora contain texts produced by learners of Russian as a foreign language. Evaluating on three diverse datasets allows for a cross-corpus comparison. We compare the performance of the minimally-supervised model to two baseline models that do not use context for candidate re-ranking, as well as to a character-level statistical machine translation system with context-based re-ranking. We show that the minimally-supervised model outperforms all of the other models. We also present an analysis of the spelling errors and discuss the difficulty of the task compared to the spelling correction problem in English.

1 Introduction
The spelling correction task has been a fundamental Natural Language Processing (NLP) problem ever since the origins of the field and has enjoyed a lot of attention in the NLP research. It is not surprising, since correcting spelling mistakes is of practical relevance for various higher-level NLP tasks and downstream applications dealing with noisy data, such as named entity recognition, dependency parsing, information retrieval, topic modeling, machine translation, essay scoring, speech recognition, automatic text correction (van Strien et al., 2020). Running a spellchecker is now a common pre-processing step performed in essay scoring (Flor, 2012a), grammatical error correction (Chollampatt and Ng, 2018; Rozovskaya and Roth, 2016; Grundkiewicz and Junczys-Dowmunt, 2018) and numerous other applications. Nevertheless, even for English, performance of spellchecking tools is not as good as one would expect, especially in noisy domains (Flor et al., 2019). Kantor et al. (2019) evaluate three publicly available spellcheckers on English learner data and find that the highest recall achieved is that of 69%, and the best precision is 57%, which indicates that the task is far from being solved. Further, their simple in-house implementation outperforms by a large margin all of the common publicly available spellcheckers.

One reason for the slow progress on the task might be the lack of common benchmark datasets. As a result, proposed methods are being evaluated either on isolated spelling errors extracted from a corpus without context or on artificially created datasets. Recently, Flor et al. (2019) released a dataset annotated for spelling errors in English learner essays and provided an evaluation of a minimal supervision system that combines features based on the misspelled word itself and the context in which it appears. They report strong performance on that corpus, as well as competitive results on a dataset from the medical domain.

We address the problem of correcting non-word spelling mistakes in Russian, a language with rich morphology. Our goal is two-fold: first, to implement various established spelling methods with known results for English and determine how they perform on Russian. Our second goal is to perform cross-corpus comparison, by evaluating on several Russian datasets that contain diverse data (texts by native Russian speakers in the social media domain, as well as Russian learner texts).

We implement four models. First, we use two baselines that do not take context into account: Aspell spellchecker and the model proposed in Kantor et al. (2019) that was shown to outperform Aspell and several other grammar checkers.

\[\text{Context in this work refers to the sentence (or the n-gram window) in which the misspelled word occurs.}\]
ers for English. We then implement two models that also take into account contextual information when proposing a correction: a statistical machine translation (SMT) approach and a minimally-supervised method. The minimally-supervised model follows the approach of Flor et al. (2019). This model is compared against a character-level SMT spellchecker that takes context into account by incorporating a word-based language model (LM) as well as other context features at the re-ranking level (Chollampatt and Ng, 2018). We evaluate these four methods on three Russian datasets that contain annotated spelling mistakes. We perform a detailed error analysis and identify the challenges pertaining to the Russian language. We show that even though spelling correction of non-word errors is considered to be an easy task, performance on a morphologically-rich language is challenging and leaves a wide gap for future research.

This paper makes the following contributions: (1) we implement and evaluate four established approaches to spelling correction and evaluate these on three Russian datasets; (2) we show that the minimally-supervised approach outperforms the other methods and is the most robust; (3) we perform error analysis identifying challenges of spelling correction for Russian.

Section 2 reviews related work on the spelling correction of Russian and on the established methods well-studied for English. Section 3 describes the three datasets of spelling errors used in this work. Section 4 presents the models. In Section 5, we present the results, and in Section 6 we perform error analysis of the results. Section 7 concludes.

2 Related Work

A non-word misspelling is a spelling error, such that the resulting string is not a valid word in the language. This is different from real-word (context-sensitive) errors, for example confusing “their”, “there” and “they’re” (Wilcox-O’Hearn et al., 2008). Context-sensitive errors also subsume grammar errors made by non-native speakers (e.g. confusing “a” and “the”), but these typically are addressed using a different set of methods (Ng et al., 2014).

Most of the spelling correction research has been focused on the English language. When dealing with a language that has rich morphology, such as Russian, specific challenges may arise. For example, the rich morphology of Russian, as we show, affects the candidate generation algorithm, where a substantially higher number of competing candidates is being generated, including those that are morphological variants of the same lemma. There is very little spelling work on other languages with complex and diverse morphology. For instance, Oflazer (1996); Mohit et al. (2014); Rozovskaya et al. (2015) address a variety of errors in Arabic, including grammar and usage errors, but they do not focus on spelling.

Previous studies on Russian spelling mainly addressed correcting spelling errors in search queries (Baytin, 2008; Panina et al., 2013), which is a special subtask of spelling correction, as the surrounding context for candidate selection is not considered or is considered in a quite restrictive way. Sorokin et al. (2016) introduced the first competition on spelling correction for Russian, which focused on correcting texts collected from Russian social media websites. Sorokin (2017) presents a follow-up study, where they show that the use of morphological information for candidate selection is beneficial for languages with well-developed morphology, such as Russian. We use the corpus released in this competition and show that it is quite different from the other two corpora used in this work.

Approaches to non-word spelling correction

Broadly speaking, the approaches to correcting non-word spelling errors can be broken down into those that only consider the characteristics of the target token when ranking correction candidates, and those that also take into account contextual information. Among the former are those that compute edit distance (Levenshtein, 1966; Damerau, 1964) and phonetic similarity between the misspelling and the candidate correction (Toutanova and Moore, 2002).

One standard approach to correcting non-word spelling errors follows the noisy channel model formulation (Shannon, 1948). This approach incorporates non-contextual information, such as the edit distance and phonetic similarity between the misspelling and the candidate correction, and the candidate frequency (Kernighan et al., 1990; Church and Gale, 1991; Toutanova and Moore, 2002). Essentially, weights for different edit operations are estimated from a large training corpus of annotated spelling errors. However, this approach requires a lot of supervision: thousands
of annotated errors paired with their corrections are used to estimate probabilities associated with each edit. While the noisy channel model can also incorporate contextual information, in general, adding new features from a variety of sources is not straightforward in the noisy channel formulation.

Flor et al. (2019); Flor and Futagi (2012) proposed a minimally-supervised model that combines contextual and non-contextual information. In Flor et al. (2019), they evaluate the model on two spelling corpora: an English learner corpus and a corpus from the biomedical domain, showing competitive results. Importantly, unlike the noisy channel model, their model only requires a small amount of supervision and is robust on out-of-domain data. In this work, we describe an implementation of this model for Russian.

SMT methods for Spelling Correction
Character-level statistical machine translation has been widely used for spelling correction of natural data as well as OCR post-correction, which can be viewed as a subtask of spelling correction. Neural network (NN) approaches, in particular, seq2seq models have recently been used for spelling correction. We do not evaluate NN methods in this work, as we have very limited amounts of training data. For an analysis and evaluation of NN approaches for spelling correction, we refer the reader to Schnober et al. (2016) and Amrhein and Clematide (2018).

3 Datasets
We use three Russian datasets annotated for misspellings. The first one, RULEC-GEC, is a learner corpus collected at the University of Oregon and consists of essays written by learners of Russian as a foreign language and heritage speakers (Alsufieva et al., 2012; Rozovskaya and Roth, 2019). The dataset was corrected and annotated by native Russian speakers and is error-coded. It is annotated exhaustively for various grammar and usage errors, and contains a large proportion of spelling errors, especially for heritage speakers (over 42% of all errors), and over 18% of all errors in the foreign group. We only focus on mistakes that are marked as spelling errors. The corpus is partitioned into training, development, and test. Since we focus on the spelling errors, we evaluate only with respect to those mistakes and ignore other annotated errors in the data.

The second corpus, henceforth RU-Lang8 (Trinh and Rozovskaya, 2021), is a dataset collected from the online language learning platform Lang-8 (Mizumoto et al., 2011) and annotated by native speakers. The dataset contains texts by learners of a variety of foreign languages. The annotation is publicly available for research. RU-Lang8 contains 54,000 tokens split up into development and test partitions. We only use the test partition in this work for evaluation, as the models are developed and tuned on the RULEC-GEC data. RU-Lang8 differs from RULEC-GEC: the latter consists of essays written on a University setting in a controlled environment, while the Lang-8 data was collected online; the majority of texts are short paragraphs or questions posed by language learners. RU-Lang8 is thus more informal and contains data by learners of multiple first language backgrounds (unlike RULEC-GEC, whose authors are from the United States).

The third corpus, RUSpellRU, is a dataset released as part of the competition on automatic spelling correction for the Russian language, which focused on social media texts. The dataset is a collection of essays from Russian blogs and social media. This is another unique dataset, very distinct: it contains a lot of colloquialisms, slang expressions and social media spelling conventions (Sorokin et al., 2016). Since the corpus contains social media texts, the misspellings include, in addition to typos, a lot of slang and colloquial forms common in social media spelling, such as the use of digits inside the words or unconventional spellings, e.g. using phonetic spelling instead of standard one.

Statistics on the datasets, including the total number of tokens as well as the spelling error rates (percentage of tokens containing a spelling error), are shown in Table 1. We observe that the RUSpellRU dataset is the most noisy one, and its error rate is more than five times higher than in the RULEC-GEC corpus, where the percentage of tokens containing a spelling mistake is the smallest among the three. On the other hand, the RUSpellRU dataset is produced by native Russian speakers, while the other two are produced by learners of Russian and thus also contain other, grammar and usage-related errors.

Table 2 analyzes the spelling errors with respect to the type of edit – replacement, split, or merge. A merge is a misspelling where a space is incor-
Table 1: Corpora statistics.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Token counts</th>
<th>Spelling errors</th>
<th>Error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>RULEC-GEC (train)</td>
<td>83,410</td>
<td>1,023</td>
<td>1.23</td>
</tr>
<tr>
<td>RULEC-GEC (dev)</td>
<td>41,163</td>
<td>497</td>
<td>1.21</td>
</tr>
<tr>
<td>RULEC-GEC (test)</td>
<td>81,693</td>
<td>1,055</td>
<td>1.30</td>
</tr>
<tr>
<td>RU-Lang8</td>
<td>31,603</td>
<td>692</td>
<td>2.19</td>
</tr>
<tr>
<td>RUSpellRU</td>
<td>28,112</td>
<td>1,963</td>
<td>6.98</td>
</tr>
</tbody>
</table>

Table 2: Distribution of annotated misspellings by type (merges, splits, replacements) in the three datasets. A merge is a misspelling where a space is incorrectly omitted, while a split is a misspelling that results from an extra space being added.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Edit type</th>
<th>Perc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Repl.</td>
<td>Merge</td>
</tr>
<tr>
<td>RULEC-GEC</td>
<td>92.4</td>
<td>1.6</td>
</tr>
<tr>
<td>RU-Lang8</td>
<td>95.7</td>
<td>1.5</td>
</tr>
<tr>
<td>RUSpellRU</td>
<td>80.2</td>
<td>11.7</td>
</tr>
</tbody>
</table>

Table 3: Distribution of annotated misspellings (replacement errors) by edit distance to correct form, in the RULEC-GEC and RU-Lang8 datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Gold errors</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RULEC-GEC</td>
<td>1055</td>
<td>65.7</td>
</tr>
<tr>
<td>RU-Lang8</td>
<td>692</td>
<td>79.9</td>
</tr>
<tr>
<td>RUSpellRU</td>
<td>1963</td>
<td>71.3</td>
</tr>
</tbody>
</table>

4 The Models

4.1 Minimally-Supervised Spelling Correction Model

We implement the model described in Flor and Festag (2012), Flor (2012a), that is evaluated in the original papers on the English learner corpus of TOEFL and GRE essays. It was also evaluated on the TOEFL-11 corpus as well as a corpus of biomedical English texts (Flor et al., 2019). Implementation of the model for Russian and its evaluation on the Russian data with its rich morphology is one of the contributions of the current work.

Finally, in Table 3 we analyze the replacement errors with respect to the edit distance between the source word and the correction. In the RULEC-GEC and RUSpellRU datasets, over 80% of replacement edits are within edit distance 1, where each type of change, including transposition errors, has a cost of 1. This analysis is consistent with findings in English corpora of misspellings (Flor et al., 2019). The RU-Lang8 corpus, however, contains a higher proportion of errors with edit distance greater than 1. Only 68.4% of errors are within edit distance of 1.

The Models

In this section, we describe the minimally-supervised model, the character-level SMT speller, and the two baselines that do not use context.

4 The Models

4.1 Minimally-Supervised Spelling Correction Model

We implement the model described in Flor and Festag (2012), Flor (2012a), that is evaluated in the original papers on the English learner corpus of TOEFL and GRE essays. It was also evaluated on the TOEFL-11 corpus as well as a corpus of biomedical English texts (Flor et al., 2019). Implementation of the model for Russian and its evaluation on the Russian data with its rich morphology is one of the contributions of the current work.

Finally, in Table 3 we analyze the replacement errors with respect to the edit distance between the source word and the correction. In the RULEC-GEC and RUSpellRU datasets, over 80% of replacement edits are within edit distance 1, where each type of change, including transposition errors, has a cost of 1. This analysis is consistent with findings in English corpora of misspellings (Flor et al., 2019). The RU-Lang8 corpus, however, contains a higher proportion of errors with edit distance greater than 1. Only 68.4% of errors are within edit distance of 1.

4 The Models

In this section, we describe the minimally-supervised model, the character-level SMT speller, and the two baselines that do not use context.
names, in particular those that are foreign names, or rare words, may be missing and would be mistakenly flagged as potential misspellings. Nevertheless, recall (detecting potential misspellings) is more important than maintaining high precision in this step. Our dictionary is based on the Yandex corpus (Borisov and Galinskaya, 2014). The corpus size is over 18 million tokens, and the resulting dictionary contains 2.3 million word types. To reduce the number of false positives, for the words not in the dictionary, we also check whether the token is recognized as the last or first name by the Mystem morphological analyzer (Segalovich, 2003) (if it appears non-capitalized in non-initial sentence position) or if the stem of the word is recognized as a known stem. The recall of the detection algorithm is shown in Table 4. The lowest recall of 65.7 is achieved on the RULEC-GEC dataset, while the highest recall is obtained on RU-Lang8 (79.9%).

We further analyze the recall of the detection algorithm by classifying the spelling mistakes in the gold data that were missed (Table 5). In the RULEC-GEC dataset, 22.8% of these errors are context-sensitive spelling mistakes, i.e. spelling errors that involve confusing valid words and which are not covered in this task. 20.6% are spelling errors that are multi-token (i.e. require merging two or more tokens), while 18.4% are context-sensitive grammar mistakes (e.g. noun case) which were miscategorized by the annotator. Another 15.4% of mistakes are capitalization errors. Only 16.9% of the missed errors (category Other) as well as 5.9% of errors that involve spelling mistakes on proper names are in fact spelling mistakes that should have been detected at this stage. Similarly, on RU-Lang8 dataset, 40.3% of missed errors are context-sensitive errors, and the actual mistakes that were missed (categories Other and Proper Name) include 33.8% of all missed tokens. In the RUSpellRU corpus, these errors comprise 8.5%.

If we exclude the non-relevant errors that are counted as missed, the recall of the detection stage improves to 89.2% for the RULEC-GEC corpus, 92.2% for the RU-Lang8 corpus, and 96.7% on the RUSpellRU dataset.

**Candidate Generation** We consider several approaches to candidate generation based on the edit distance between the source and the target strings. Candidates are generated using the dictionary described in the previous section. Candidates include all dictionary words within edit distance that does not exceed half the length of the misspelled string; the maximum distance is set to three, as the number of candidates grows very quickly due to the rich morphology of Russian (see Table 6). For example, on average, 3 candidates are generated with edit distance of 1 for RULEC-GEC. This number increases to 313 when an edit distance of 3 is used instead. This is because, due to the morphological complexity, morphological variants of the same base word are included as different candidates (also discussed in Section 6). In English, candidates up to edit distance of 6 are included (Flor et al., 2019), but doing so would explode the search space.

The candidate generation algorithm is evaluated

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Dist.</th>
<th>Cand. per error in cand. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RULEC-GEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3.1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>44.3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>313.0</td>
</tr>
<tr>
<td>RU-Lang8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4.6</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>76.0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>412.6</td>
</tr>
<tr>
<td>RUSpellRU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3.7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>70.6</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>361.9</td>
</tr>
</tbody>
</table>

Table 6: Evaluation of the candidate generation step.
<table>
<thead>
<tr>
<th>Feature name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Non-contextual features</strong></td>
<td></td>
</tr>
<tr>
<td>Orthographic similarity</td>
<td>Inverse edit distance</td>
</tr>
<tr>
<td>Character difference</td>
<td>A pair comprising original and replacement character</td>
</tr>
<tr>
<td>Candidate frequency</td>
<td>Unigram word frequency</td>
</tr>
<tr>
<td><strong>Contextual features</strong></td>
<td></td>
</tr>
<tr>
<td>N-gram support</td>
<td>N-gram counts in the 4-word window</td>
</tr>
</tbody>
</table>

Table 7: Description of all the features used in candidate ranking with the minimally-supervised model.

in Table 6. As the edit distance increases, the recall of the candidate generation (i.e. proportion of errors for which gold is among the generated candidates) improves, however, the number of candidates per error increases exponentially. We note, though, that, while on the RULEC-GEC and RU-Lang8 datasets, the recall increases to over 90% with the edit distance set to 3, on the RUSpellRU corpus, the highest recall achieved is 78.8%, even though 83.6% of misspellings are within edit distance of 1, as shown in Table 3. This indicates that a large number of colloquial and slang words present in the corpus are not found in the dictionary.

**Ranking of Candidate Corrections** The ranking step is the most challenging one and is the focus of most work on non-word spelling correction (Fivez et al., 2017). Ranking of correction candidates in the minimally-supervised model uses both the features of the misspelling-candidate pair and the contextual information. Flor (2012b) tuned feature weights manually on a set of misspellings, extracted from a corpus of TOEFL and GRE essays. In this work, similar to (Flor et al., 2019), feature weights are learned using a linear algorithm (Averaged Perceptron (Rosenblatt, 1958), implemented within Learning Based Java (Rizzolo and Roth, 2007).

We implement the following features: orthographic similarity (inverse edit distance), character-difference, candidate word frequency, and n-gram support. The features are listed in Table 7 and described below.

**Orthographic similarity** is computed as inverse edit distance, $1/(eDist + 1)$, where $eDist$ is the edit distance (including transpositions) between the misspelling and the correction candidate (Levenshtein, 1966; Damerau, 1964).

**Character difference** is a feature that encodes the specific letter change between the original and the candidate. This feature is active with replacement, deletion, and character insertion errors for candidates whose edit distance is 1. The feature is expected to reflect some common and well-known character confusions, both among native and non-native Russian writers, e.g. omitting the “s” at the end of a word after character “ц” or incorrectly using a instead of o in an unstressed position. Note that this feature is similar to the concept of encoding phonetic similarity, which we omit in this implementation.

**Candidate frequency** A more frequent word is more likely to be the intended word than a rare word (Flor, 2012a). Unigram word frequency is computed for each correction candidate using the Yandex corpus.

**N-gram support** For each correction candidate, all n-grams in the window of four context words on each side are taken into account by the n-gram support feature. We use co-occurrence counts computed from a large corpus collected over the Web (235 million tokens), henceforth the Sharoff corpus. The n-gram support feature is a summation over the counts of all n-grams of length 2 to 5 (excluding the unigram count of the candidate itself, since its frequency is reflected in the candidate frequency feature). For each error, the n-gram count value is normalized by the highest candidate count for that error.

For each misspelled token, with the exception of the letter difference feature, the feature scores of its candidate corrections are normalized, by dividing the score of the candidate feature by the highest-scoring candidate on that given feature.

**4.2 The SMT Speller**

We implement a character-level statistical machine translation (SMT) speller (Chollampatt and Ng, 2017). Input to the character-level SMT component is a sequence of characters that make up the unknown (misspelled) word and output is a list of correction candidates (words). In Chollampatt and Ng (2017), the misspelled words are those words that have not been observed in the source side of the parallel training data used to train the translation model. In this work, the unknown words are identified using the same detection algorithm described in Section 4.1. We do this for two reasons: first, due to lack of large amounts of parallel data

---

The corpus was kindly shared by Serge Sharoff.
and the morphological complexity of Russian, the number of unknown words for a word-level SMT system would be too high. Second, we wish to keep the detection step fixed, which allows for a fair comparison of the re-ranking algorithms.

The character-level translation model, in line with Chollampatt and Ng (2017), is trained on pairs of misspellings and their corrections from the RULEC-GEC training corpus (774 pairs) and an additional set of 1,000 correct words selected uniformly at random from the target side of RULEC-GEC training data. The language model that is part of the SMT system is a 5-gram character-level model trained on the Yandex corpus (22 million tokens). The SMT model is tuned on the misspelling-correction pairs from the RULEC-GEC development set. The character-level SMT model is tuned using MERT (minimum error-rate training) on characters, with character-level edit operation features and a 5-gram character LM.

For each unknown word, the character-level SMT produces 100 candidates that are then rescored to select the best candidate based on the context. The rescoring is done following Chollampatt and Ng (2017) and uses word-level n-gram LM features: LM probability and the LM OOV (out-of-vocabulary) count denoting the number of words in the sentence that are not in the LM’s vocabulary. The word-level n-gram LM is trained on the Sharoff corpus, using the KenLM toolkit (Heafield et al., 2013).

### 4.3 Further Baseline Systems

We compare to two other methods that do not make use of context information: Aspell, and a re-ranking approach proposed in Kantor et al. (2019). The latter has been recently used in grammar and spelling correction research and showed good results in English.

**Kantor et al. re-ranking** Kantor et al. (2019) implement an approach to English spelling correction, that is quite simple but is surprisingly effective and outperforms substantially other commonly used open-source spellcheckers: Enchant, Norvig, and Jamspell. Briefly, the approach relies on a large dictionary compiled from a native corpus to identify misspelled tokens. In re-ranking, for each misspelling, the most frequent candidate correction (with a minimum count of 20) within an edit distance of 1 (transposition is treated as a distance of 1) is returned. If no such candidate exists, they check if the misspelled word can be split into two words that are in the word-count data or in the dictionary. They implement their re-ranking method (keeping the minimum count for words at 5, since Russian is a morphologically-rich language). Our list of incorrect tokens is generated using the same candidate detection step described above (Section 4.1). Only the re-ranking is different.

### 5 Results

In all cases, the models are trained on the RULEC-train corpus and tuned on the RULEC development data. All results are reported on the test partitions of the three datasets. Key results of the minimally-supervised model on the three datasets are shown in Table 8. We observe that the performance on RU-Lang8 is significantly lower than on the other two datasets. We conjecture that this may be due to the fact that RU-Lang8 has a small proportion of errors with corrections being within an edit distance of 1 from the misspelled token (see Table 3).

The results of the minimally-supervised model and of the other models implemented in this work are shown in Table 9. The minimally-supervised model outperforms all of the other models significantly on all three datasets. The relative performance of the models on each dataset is consistent: we note that Aspell has the poorest performance. This is followed by the SMT approach and the approach by Kantor et al. (2019). The two approaches are quite close, although the SMT method has high precision on RULEC-GEC and the RUSpellRU datasets. The minimally-supervised method achieves a substantially higher recall than all the other methods. It also achieves the highest precision on RULEC-GEC and RUSpellRU, although on the RU-Lang8 corpus its pre-

<table>
<thead>
<tr>
<th>Dataset</th>
<th>P</th>
<th>R</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>RULEC-GEC</td>
<td>71.6</td>
<td>46.5</td>
<td>64.6</td>
</tr>
<tr>
<td>RU-Lang8</td>
<td>54.0</td>
<td>42.7</td>
<td>51.3</td>
</tr>
<tr>
<td>RUSpellRU</td>
<td>74.5</td>
<td>59.0</td>
<td>65.9</td>
</tr>
</tbody>
</table>

Table 8: Key results of the minimally-supervised model. Performance on the error correction, using the full set of features. The model is trained on the RULEC-train corpus. Since edit distance 1 is used, this feature is omitted.
### Table 9: Comparison of the minimally-supervised model with other systems implemented in this work.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>System</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RULEC-GEC</td>
<td>Aspell</td>
<td>42.2</td>
<td>41.2</td>
<td>42.0</td>
</tr>
<tr>
<td></td>
<td>Kantor et al. (2019)</td>
<td>65.5</td>
<td>43.6</td>
<td>59.6</td>
</tr>
<tr>
<td></td>
<td>SMT</td>
<td>70.1</td>
<td>34.4</td>
<td>58.1</td>
</tr>
<tr>
<td></td>
<td>Minim.-super.</td>
<td>71.5</td>
<td>46.5</td>
<td>64.6</td>
</tr>
<tr>
<td>RU-Lang8</td>
<td>Aspell</td>
<td>33.8</td>
<td>7.3</td>
<td>19.6</td>
</tr>
<tr>
<td></td>
<td>Kantor et al. (2019)</td>
<td>50.1</td>
<td>41.9</td>
<td>48.2</td>
</tr>
<tr>
<td></td>
<td>SMT</td>
<td>49.9</td>
<td>28.5</td>
<td>43.4</td>
</tr>
<tr>
<td></td>
<td>Minim.-super.</td>
<td>42.7</td>
<td>54.0</td>
<td>51.3</td>
</tr>
<tr>
<td>RUSpellRU</td>
<td>Aspell</td>
<td>34.2</td>
<td>35.8</td>
<td>35.0</td>
</tr>
<tr>
<td></td>
<td>Kantor et al. (2019)</td>
<td>59.5</td>
<td>48.4</td>
<td>53.4</td>
</tr>
<tr>
<td></td>
<td>SMT</td>
<td>66.7</td>
<td>20.1</td>
<td>44.9</td>
</tr>
<tr>
<td></td>
<td>Minim.-super.</td>
<td>74.5</td>
<td>59.0</td>
<td>65.9</td>
</tr>
</tbody>
</table>

### Table 10: Evaluation of different edit distances in candidate generation. Performance on error correction of the minimally-supervised model, using the full set of features. The model is trained on RULEC-train.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Edit dist.</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RULEC-GEC</td>
<td>1</td>
<td>71.6</td>
<td>46.5</td>
<td>64.6</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>67.5</td>
<td>51.0</td>
<td>63.4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>62.1</td>
<td>50.5</td>
<td>61.0</td>
</tr>
<tr>
<td>RU-Lang8</td>
<td>1</td>
<td>42.7</td>
<td>54.0</td>
<td>51.3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>46.5</td>
<td>51.8</td>
<td>50.6</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>49.1</td>
<td>53.6</td>
<td>52.7</td>
</tr>
<tr>
<td>RUSpellRU</td>
<td>1</td>
<td>74.5</td>
<td>59.0</td>
<td>65.9</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>67.5</td>
<td>59.9</td>
<td>63.4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>65.4</td>
<td>59.5</td>
<td>62.3</td>
</tr>
</tbody>
</table>

### Table 11: Feature ablation. Performance on the error correction, using an edit distance of 1. The model is trained on the RULEC-train corpus.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Counts</th>
<th>P</th>
<th>R</th>
<th>F-s</th>
</tr>
</thead>
<tbody>
<tr>
<td>RULEC-GEC</td>
<td>all feats</td>
<td>71.5</td>
<td>46.5</td>
<td>64.6</td>
</tr>
<tr>
<td></td>
<td>no cand freq</td>
<td>71.2</td>
<td>46.1</td>
<td>64.2</td>
</tr>
<tr>
<td></td>
<td>no char. diff</td>
<td>71.2</td>
<td>46.2</td>
<td>64.2</td>
</tr>
<tr>
<td></td>
<td>no n-gram</td>
<td>67.5</td>
<td>43.9</td>
<td>61.0</td>
</tr>
<tr>
<td>RU-Lang8</td>
<td>all feats</td>
<td>42.7</td>
<td>54.0</td>
<td>51.3</td>
</tr>
<tr>
<td></td>
<td>no cand freq</td>
<td>43.5</td>
<td>55.1</td>
<td>52.3</td>
</tr>
<tr>
<td></td>
<td>no char. diff</td>
<td>43.2</td>
<td>54.7</td>
<td>52.0</td>
</tr>
<tr>
<td></td>
<td>no n-gram</td>
<td>40.2</td>
<td>51.1</td>
<td>48.5</td>
</tr>
<tr>
<td>RUSpellRU</td>
<td>all feats</td>
<td>74.5</td>
<td>59.0</td>
<td>65.9</td>
</tr>
<tr>
<td></td>
<td>no cand freq</td>
<td>74.0</td>
<td>58.7</td>
<td>65.4</td>
</tr>
<tr>
<td></td>
<td>no char. diff</td>
<td>74.4</td>
<td>58.9</td>
<td>65.7</td>
</tr>
<tr>
<td></td>
<td>no n-gram</td>
<td>72.1</td>
<td>57.1</td>
<td>63.8</td>
</tr>
</tbody>
</table>

### Evaluation of different edit distance values in candidate generation

Next, we evaluate performance as a function of edit distance values for candidate generation. In all cases, we use the full feature set. Results are shown in Table 10. We observe that there is no clear benefit to using an edit distance greater than 1: on the RULEC-GEC corpus, recall slightly improves, while precision drops. On the RU-Lang8 dataset, precision improves with a larger edit distance, while recall remains the same. On the RUSpellRU dataset, recall does not change, while precision drops. For this reason, we use an edit distance of 1 in candidate generation, as the number of candidates is much smaller, as discussed above. It should be noted that, while using only edit distance 1 is optimal for the scoring with the system, it is not sufficient, as there are still many misspellings in each of the datasets (as shown in Table 3) that have corrections within higher edit distances. We leave this for future work.

### Feature ablation

Finally, we perform feature ablation to evaluate the contribution of the various features in the minimally-supervised model. Results are shown in Table 11. The n-gram support feature is shown to be the most important: dropping this features results in performance loss of 2-3 points on each dataset. This result demonstrates the significance of the contextual information in spelling correction.

### 6 Error Analysis

#### Candidate re-ranking

We perform error analysis of the candidate re-ranking component that is part of the minimally-supervised approach. From each dataset, we analyze 100 errors, on which an incorrect candidate is preferred. Results are shown in Table 12. *Morph. variant* refers to incorrect candidates that is an inflectional morphological variant of the correct suggestion. *Wrong cand.* denotes an incorrect suggestion that is not morphologically related to the correct suggestion. *Dist.* denotes corrections that have an edit distance greater than 1 to the source word. Since we currently only consider candidates within edit distance of 1, these errors cannot be corrected. *Lex. change* refers to misspellings that are also word usage errors. It is interesting to note that all three corpora have the same proportion of errors (50%) that could not be corrected.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Morph. variant</th>
<th>Wrong cand.</th>
<th>Dist.</th>
<th>Lex. change</th>
</tr>
</thead>
<tbody>
<tr>
<td>RULEC-GEC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RU-Lang8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUSpellRU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 12: Distribution of incorrect suggestions by the candidate re-ranking algorithm. Morph. variant refers to an incorrect candidate that is a morphological variant of the correct suggestion. Wrong cand. denotes an incorrect suggestion that is not morphologically related to the correct suggestion. Edit dist. denotes corrections that have an edit distance greater than 1 to the source word. Lex. change are errors that are word usage errors, in addition to having a spelling error.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Morph.</th>
<th>Wrong</th>
<th>Edit</th>
<th>Lex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RULEC-GEC</td>
<td>23</td>
<td>13</td>
<td>50</td>
<td>14</td>
</tr>
<tr>
<td>RU-Lang8</td>
<td>17</td>
<td>13</td>
<td>50</td>
<td>20</td>
</tr>
<tr>
<td>RU-SpellRU</td>
<td>20</td>
<td>31</td>
<td>49</td>
<td>0</td>
</tr>
</tbody>
</table>

corrected due to the edit distance of the correct candidate being greater than 1. Further, the corpora have similar distributions overall regarding mistakes when selecting a morphological variant (17-23%). This shows that for languages with rich morphology, morphological variants present an issue for spelling correction, since morphological variants typically differ by one character change, and thus correction candidates typically include multiple morphological variants of the same word. A similar conclusion, although not quantified, was drawn in the RUSpellRU competition for the social media data (Sorokin et al., 2016). We confirm this finding for various corpora and quantify it. Both of the learner corpora also have grammatical errors, and some of these were mistagged as spelling mistakes (14% and 20% in the RULEC and RU-Lang8 corpora, respectively). In contrast, because the RUSpellRU corpus contains data from native speakers, it is not expected to have many grammar-related errors.

7 Conclusion

In this paper, we implement four models for spelling correction for Russian and evaluate these on three diverse datasets that contain spelling mistakes. We present a comparative analysis of spelling mistakes contained in the three datasets. Evaluation results show that the minimally-supervised model outperforms two baseline models that do not use context when selecting a candidate correction, and another model that uses a character-level SMT and a language model in re-ranking. We perform feature ablation of the minimally-supervised model showing that contextual information contributes to the performance. We also carry out error analysis that reveals that one common source of errors in Russian in selecting the appropriate correction candidate is the presence of morphological variants. This study should provide insight into the spelling correction problem for languages with rich morphology.

Acknowledgments

We thank the three anonymous reviewers for their insightful comments.

References


1219


Abstract

In translating text where sentiment is the main message, human translators give particular attention to sentiment-carrying words. The reason is that an incorrect translation of such words would miss the fundamental aspect of the source text, i.e. the author’s sentiment. In the online world, MT systems are extensively used to translate User-Generated Content (UGC) such as reviews, tweets, and social media posts, where the main message is often the author’s positive or negative attitude towards the topic of the text. It is important in such scenarios to accurately measure how far an MT system can be a reliable real-life utility in transferring the correct affect message. This paper tackles an under-recognised problem in the field of machine translation evaluation which is judging to what extent automatic metrics concur with the gold standard of human evaluation for a correct translation of sentiment. We evaluate the efficacy of conventional quality metrics in spotting a mistranslation of sentiment, especially when it is the sole error in the MT output. We propose a numerical ‘sentiment-closeness’ measure appropriate for assessing the accuracy of a translated affect message in UGC text by an MT system. We will show that incorporating this sentiment-aware measure can significantly enhance the correlation of some available quality metrics with the human judgement of an accurate translation of sentiment.

1 Introduction

Standard quality measures for assessing the performance of machine translation systems, such as BLEU (Papineni et al., 2002), are domain agnostic; they evaluate the translation accuracy regardless of the semantic domain or linguistic peculiarities of the source text. Consequently, they give equal penalty weight to inaccurate translation of n-grams, which may lead to performance overestimation (or underestimation). For example, the Arabic high-rated Goodreads book review showing the reviewers overall satisfaction of a novel: ‘الرواية رهيبة عيبها الوحيد الجزء الآخير’ (‘The novel is great, its only flaw is the last part’) is mistranslated by the available online translation tool which wrongly outputs a negative sentiment (‘The novel is terrible, its only flaw is the last part’). Despite the distortion of the affect message, this translation receives an equally high score as a correct translation (‘The story is great, its only flaw is the last part’), but which uses story instead of novel. This is because BLEU mildly penalises the wrong translation as it swaps only one uni-gram (‘great’) with its opposite (‘terrible’). Yet, the mistranslation of this particular uni-gram is a critical error as it is most pivotal in transferring the sentiment, and hence the MT performance is over-estimated.

There have been numerous efforts to address the common pitfalls of n-gram-based evaluation metrics by incorporating semantic and contextual features. However, despite research evidence of its analytical limitations, BLEU is still the de facto standard for MT performance evaluation (Mathur et al., 2020a; Reiter, 2018). Moreover, although the introduction of more semantically-oriented metrics showed a better correlation with human judgement, still the estimation of sentiment preservation in UGC has not yet been investigated. This is par-
tially due to the fact that the domain and linguistic style of the WMTs datasets typically used for metric evaluation (e.g. newstest2020, Canadian Parliament, Wikipedia, UN corpus) is quite different than the non-standard noisy UGC where sentiment is the main content of a telegraphic message (Ma et al., 2018; Barrault et al., 2019; Mathur et al., 2020b). Assessed over these WMT datasets, some metrics manifest an almost perfect correlation with human evaluation on the segment-level (e.g. WMT20 participating metrics record results of up to 0.97 Pearson correlation on the newstest2020) (Mathur et al., 2020b). However, research has also shown that metrics usually report weaker correlation with low human assessment score ranges (Takahashi et al., 2020a,b).

In this paper, we argue that the high correlation of some metrics may not be replicated with a different domain such as sentiment-oriented UGC, specifically when there is a mistranslation of sentiment-critical word(s). For this reason, we propose a ‘sentiment-closeness’ measure that can accommodate for a better evaluation of the MT system’s ability to capture the correct sentiment of the source text.

To present our sentiment quality measure, we first briefly explain in Section 2 why metrics commonly used for MT quality estimation may not always be efficient in assessing sentiment-critical translation errors. In Section 3, we present an experiment to quantitatively assess the divergence of the analysed metrics from a human judgement of a correct/incorrect translation of an affect message. Section 4 presents our proposed solution for fine-tuning quality metrics for a better correlation with human judgement. Section 5 presents the results of incorporating our sentiment-measure in different quality metrics. In Section 6 we conduct an error analysis of our empirical approach and discuss its limitations. Finally, Section 7 presents a conclusion on our conducted experiments.

2 Available Metrics

Automatic evaluation metrics usually take the output of an MT system (hypothesis) and compare it to one or several translations produced by human translator(s) (reference). Based on their matching methods, the most commonly used automatic metrics can be broadly categorised into: surface n-gram matching and embedding matching. Surface n-gram methods work by calculating exact matching, heuristic matching or an edit distance between the aligned n-grams of the reference and hypothesis translation(s). The embedding methods, on the other hand, calculate a similarity score between learned token representations, such as contextual embedding vectors, with or without the aid of external linguistic tools. In the following sections, we briefly explain the methods behind three canonical metrics as representative of each category. We illustrate why the theoretical foundation of each metric may not be optimum for evaluating the translation of sentiment-oriented UGC.

2.1 Surface N-gram Matching Metrics

BLEU The standard metric for assessing empirical improvement of MT systems is BLEU (Papineni et al., 2002). Simply stated, the objective of BLEU is to compare n-grams of the candidate translation with n-grams of the reference translation and count the number of matches; the more the matches, the better the candidate translation. The final score is calculated using a modified n-gram precision multiplied by a brevity penalty so that a good candidate translation would match the reference translation in length, in word choice, and in word order. The disadvantage of the BLEU metric which is relevant to our present study is that it treats all n-grams equally. Due to its restrictive surface n-gram matching, it does not account for the semantic importance of an n-gram in the context of a text. Accordingly, BLEU would incorrectly give a high score to an MT output if it scores exact match with the reference except for one uni-gram, even if this uni-gram completely changes the sentiment of a text (e.g. ‘terrible’ and ‘great’ as in the Goodreads example above). Online built-in MT tools have been shown to frequently transfer the exact opposite sentiment word for some dialectical expressions in UGC translated into English (Saadany and Orasan, 2020). Therefore, the BLEU evaluation of an MT performance would be misleadingly over-permissive in such cases where only one or two sentiment-critical words are mistranslated.

METEOR METEOR (Banerjee and Lavie, 2005) incorporates semantic information as it evaluates translation by calculating either exact match, stem match, or synonymy match. For synonym matching, it utilises WordNet synsets (Pedersen et al., 2004). More recent versions (METEOR 1.5 and METEOR++2.0) apply also importance
weighting by giving smaller weight to function words (Denkowski and Lavie, 2014; Guo and Hu, 2019). The METEOR score ranges from 0 (worst translation) to 1 (best translation). There are two shortcomings to the METEOR metric which do not make it a robust solution for evaluating sentiment transfer. First, the synonym matching is limited to checking whether the two words belong to the same synset in WordNet. However, WordNet synonymy classification is different than regular thesauruses. For example, ‘glad’ is a synset of ‘happy’ and hence considered a synonym, whereas ‘cheerful’ is not a direct synset to ‘happy’ and hence would be considered a mismatch by METEOR. The following two examples illustrate further the limitations of using WordNet synonymy by METEOR:

**Example 1**
Scores: [METEOR: 0.46]
- Hypothesis: “The weather is sunny, what a happy day”
- Reference: “The sun is shining, what a cheerful day”

**Example 2**
Scores: [METEOR: 0.48]
- Hypothesis: “I’m not sure why, but I feel so happy today”
- Reference: “I don’t get it, but I feel so sad today”

The METEOR scores for Examples 1 and 2 clearly diverge from a human evaluation of a good translation. In the first example, although the translation conveys the correct emotion (‘happiness’), it receives a similar METEOR score to the hypothesis in the second example which gives the exact opposite emotion of the source (‘happiness’ instead of ‘sadness’). The inadequate scoring is a result of the WordNet taxonomy which causes the metric to equally treat both pairs (‘sad’, ‘happy’) and (‘cheerful’, ‘happy’) as non-synonym synsets and hence unmatched chunks.

The second problem with METEOR which may affect its efficacy in evaluating sentiment transfer relates to its weighting schema for function and non-function words. The following example clarifies the gravity of this problem:

**Example 3**
Scores: [METEOR: 0.92]
- Hypothesis: “If he had blown himself up in your country, God would forgive him”
- Reference: “If he had blown himself up in your country, God would not forgive”.

The hypothesis in Example 3 is the translation output of Twitter’s built-in MT system for an Arabic tweet commenting on a terrorist attack¹. The MT failure to translate the negation marker flips the sentiment of the author from ‘anger’ against the terrorist to ‘sympathy’. Despite this, the METEOR score is 0.92 which is within the highest upper bound, ranking it as a good translation. On the other hand, the METEOR score for a correct translation of sentiment with a negation marker is 0.93. The main culprit for this inaccurate scoring is the lexical weighting which causes the metric not to penalise the missing of a negation marker which produces a sentiment-critical error. Due to the grave consequences of such mistranslations, it becomes critical to have a sentiment-sensitive metric that is capable of spotting similar errors.

### 2.2 Embedding-based Metrics

BERTScore Recently embedding-based metrics have proven to achieve the highest performance in recent WMT shared tasks for quality metrics (e.g. Sellam et al. (2020); Lo (2020); Mukherjee et al. (2020)). We take BERTScore as a representative metric for this approach (Zhang et al., 2019). BERTScore computes a score based on a pair wise cosine similarity between the BERT contextual embeddings of the individual tokens for the hypothesis and the reference (Devlin et al., 2018). Accordingly, a BERTScore close to 1 indicates proximity in vector space and hence a good translation. The main problem with embedding-based metrics, such as BERTScore, is that antonyms contain similar distributional information since they usually occur in similar contexts. Example 4 illustrates this point:

**Example 4**
Scores: [BERTScore: 0.85]
- Hypothesis: “What is this amount of anger, I don’t understand!”

¹https://twitter.com/gaston810/status/673950532340465664
Example 4 shows the mistranslation produced by Twitter’s Translate Tweet tab of an Arabic tweet. Although the sentiment polarity is flipped in the candidate translation above, the hypothesis receives a BERTScore of 0.85 which indicates a high cosine similarity to the reference in vector space and hence a good translation. Clearly, the metric score is not comparable to a human perception of the emotion reflected by the source.

Figure 1 illustrates the reason behind this misleadingly high score. Figure 1 is a 2-D visualisation of BERT’s contextual embedding vectors for the hypothesis translation (in blue) and the reference (in red). Both sentences are very close in the embedding space due to the exact match of their individual tokens. The only mismatch is between the antonyms ‘happiness’ and ‘anger’. As shown in the figure, the pre-trained embedding vectors of the opposite polarity nouns are also quite close because of their common contextual information. An embedding metric such as BERTScore, therefore, may not penalise antonyms which typically occur in similar contexts.

Recently, there have been different approaches to overcome some distributional problems of contextual embeddings. Reimers and Gurevych (2019), for example, introduce SBert, a modification of the pretrained BERT network, which should mitigate the antonymy problem. They use Siamese network structures where the embeddings of similar sentence pairs are independently learned via two parallel transformer architectures. We measured how far this technique could solve the opposite sentiment problem by measuring the SBert sentence similarity of the hypothesis and reference in Example 4. The cosine similarity of the SBert sentence embedding vectors for the hypothesis and reference in Example 4 reached 0.61. A correct translation of the reference, however, such as ‘What is all this cheerfulness, I don’t understand’ has a cosine similarity score of 0.79. This small similarity difference (0.18) would be misleading if taken as an evaluation of how far the sentiment poles in the two hypotheses are different. A more sentiment-targeted measure is needed for assessing mistranslations due to this antonymy problem.

In the following section, we conduct an experiment to quantify the divergence of the above mentioned metrics from the human perception of a proper sentiment transfer in a translated text.

3 Evaluation of Mistranslated Sentiment

We have shown in the previous section that three canonical MT evaluation metrics do not give a penalty proportional to sentiment-critical errors on segment-level by an MT online tool. In order to quantify how far the aforementioned quality metrics diverge or correlate with human judgement of sentiment transfer, we measure the performance of each metric on a dataset of tweets that had sentiment-critical translation errors. To compile this data, first we used the Twitter built-in translation system (Google API) to translate a dataset of tweets annotated for sentiment. The source dataset amounted to $\approx 7,000$ tweets in three languages: English, Arabic and Spanish. We translated the Arabic and Spanish into English and the English was translated into Spanish and Arabic.

To extract instances where the MT system failed to translate the sentiment correctly, we built an English sentiment-detection classifier by fine-tuning a Roberta XML model (Liu et al., 2019) on an English dataset of 23,000 tweets annotated for sentiment. The English classifier was used to predict the sentiment of the Google API output for the translation of the Arabic and Spanish tweets into English and the English back-translation of the English tweets translated into Spanish and Arabic.

The classifier’s predicted sentiment was compared to the gold standard emotion of the source text, and
instances of discrepancy were extracted as potential mistranslations of sentiment. Finally, from the extracted instances, we manually built a translation quality evaluation dataset.

The quality evaluation dataset, henceforth QE, consisted of target tweets where the error is exclusively a mistranslation of the sentiment-carrying lexicon. In these tweets, the mistranslations either completely flip the sentiment polarity of the source tweet, similar to Examples 3 and 4 above, or transfer the same polarity but with a mitigated sentiment tone. The tweets with exclusive sentiment translation errors amounted to 300 tweets. We also added 100 tweet/translation pairs where the MT system transfers the correct sentiment. Reference translations of the QE dataset were created by native speakers of the respective source languages. Essentially, the reference translations aimed at correcting chunks that caused a distortion of the affect message and retained as many of the hypothesis n-grams as possible to detect how far each metric is sensitive to sentiment mismatching and not to the mismatch in other non-sentiment carrying words. The translators were also asked to assign a score to each pair of source-hypothesis tweet, where 1 is the poorest sentiment transfer and 10 is best sentiment transfer. The average scores of annotators were taken as the final human score.

To quantify the ability of the three metrics explained in Section 2 to assess the transfer of sentiment in the QE dataset, we compared their scores of the translation hypotheses with the human judgement scores for sentiment transfer on the segment-level. We followed the WMT standard methods for evaluating quality metrics and used absolute Pearson correlation coefficient $r$ and the Kendall correlation coefficient $|\tau|$ to evaluate each metric’s performance against the human judgement. Figures 2 and 3 show heatmaps visualising the Pearson and Kendall correlation coefficients for the studied metrics and the human scores, respectively.

As seen from Figures 2 and 3, both BERTScore and METEOR achieve a better correlation with the human judgement than BLEU which achieves only 0.16 and 0.12 Pearson and Kendall correlations, respectively. However, the relatively overall low correlations (max $r = 0.39$ and max $|\tau| = 0.27$) raise important doubts as to the reliability of these accepted metrics for ranking MT systems which translate sentiments. Bearing in mind that 75% of the segments in the QE dataset have critical translation errors that seriously distort the sentiment, the low correlation results highlight the need for a sentiment-targeted measure that can improve a metric’s efficacy in capturing mistranslated sentiment by an MT system in real-life scenarios.

4 Sentiment-Aware Measure (SAM) for Machine Translated UGC

In this section, we propose a new measure for assessing MT performance that takes into account the sentiment similarity between the MT system translation and the reference. This sentiment measure should be used as a fine-tuning tool to adjust a quality score in cases where it is used to assess the translation quality of sentiment-oriented text. The SAM score is calculated by using the SentiWord dictionary of prior polarities (Gatti et al., 2015). SentiWord is a sentiment lexicon that combines the high precision of manual lexica and the high coverage of automatic ones (covering 155,000 words). It is based on assigning a ‘prior polarity’ score for each lemma-POS in both SentiWordNet and a number of human-annotated sentiment lexica (Baccianella et al., 2010; Warriner et al., 2013).
prior polarity is the out-of-context positive or negative score which a lemma-POS evokes. It is reached via an ensemble learning framework that combines several formulae where each lemma-POS is given the score that receives the highest number of votes from the different formulae. SentiWord prior polarity scores have been proven to achieve the highest correlation with human scores in sentiment analysis regression and classification SemEval tasks (Gatti et al., 2015).

We assume that our sentiment adjustment factor, SAM, is proportional to the distance between the sentiment scores of the unmatched words in the system translation and the reference, the higher the distance the greater the SAM adjustment. To calculate the SAM score, we designate the number of remaining mismatched words in the system translation and reference translation by \( m \) and \( n \), respectively. We calculate the total SentiWord sentiment score for the lemma-POS of the mismatched words in the translation and reference sentences using a weighted average of the sentiment score of each mismatched lemma-POS. The weight of a hypothesis mismatched word \( w_h \) and a reference mismatched word \( w_r \) is calculated based on the sentiment score of its lemma-POS, \( s \), as follows:

\[
\begin{align*}
    w_h^i &= |s_i|, \quad i = 1, 2, \ldots, m. \\
    w_r^i &= |s_i|, \quad i = 1, 2, \ldots, n.
\end{align*}
\]

Then the total sentiment score for hypothesis \( S_h \) and reference \( S_r \) is given by:

\[
\begin{align*}
    S_h &= \sum_{i=1}^{m} \alpha_i s_i, \quad \alpha_i = \frac{w_h^i}{\sum_{i=1}^{m} w_h^i} \\
    S_r &= \sum_{i=1}^{n} \beta_i s_i, \quad \beta_i = \frac{w_r^i}{\sum_{i=1}^{n} w_r^i}
\end{align*}
\]

The normalised SAM adjustment is given by:

\[
p = \frac{|S_r - S_h|}{2}
\]

and the translation final score will be given by:

\[
\text{Score} = C_{hr} (1 - p)
\]

where \( C_{hr} \) can be any metric’s matching score between a translation hypothesis and a reference segment. For this experiment, we will measure \( C_{hr} \) as the BLEU, METEOR and BERTScore scores. To illustrate how SAM score adjusts a metric score with respect to the transfer of sentiment, table 1 shows the SAM score adjustment for Examples 3 and 4 above.

As can be seen from Table 1, the metric score \( C_{hr} \) (here METEOR and BERTScore, respectively) is significantly reduced due to the sentiment distance between the mismatched words \( (w_h, w_r) \) as well as their mismatched weights \( (S_h, S_r) \). The reduced scores are more representative of the distortion of sentiment produced by the MT system in Examples 3 and 4. The SAM adjustment, therefore, is a targeted-measure that can fine-tune a metric according to the ‘sentiment-closeness’ of the translation and the reference.

5 Assessing SAM performance

To check how far the SAM measure can improve the evaluation of sentiment, we assessed the performance of the three chosen metrics on the QE dataset utilised in the experiment in Section 3 with the SAM adjustment. Figures 4 and 5 show heatmaps of the Pearson and Kendall correlations of the SAM adjusted metrics with the human judgement in the QE dataset.

Compared to metric scores without the SAM adjustment, Figures 4 and 5 show that the combination of SAM with the three metrics consistently leads to an improvement in each metric’s correlation with the human judgement of a correct sentiment transfer in our dataset. Overall, BERTScore records the highest Pearson correlation coefficients and Kendall rank dependence (0.59 and 0.44, respectively). This means that it is better able to penalise critical translation errors in our QE dataset. Moreover, compared to their scores without SAM,

\[\text{Figure 4: Absolute Pearson correlations with SAM Adjustment for the QE dataset}\]

More examples of SAM adjustments are in Appendix A.
Table 1: Calculating the SAM adjustment for Examples 3 and 4

<table>
<thead>
<tr>
<th>Example</th>
<th>$w_h$</th>
<th>$w_r$</th>
<th>$S_h$</th>
<th>$S_r$</th>
<th>$p$</th>
<th>$C_{hr}$</th>
<th>Score+SAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>him#a</td>
<td>not#r</td>
<td>0</td>
<td>-1.0</td>
<td>0.5</td>
<td>0.92</td>
<td>0.46</td>
</tr>
<tr>
<td>4</td>
<td>anger#n</td>
<td>happiness#n</td>
<td>-0.669</td>
<td>0.856</td>
<td>0.762</td>
<td>0.85</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Figure 5: Kendall correlations with SAM adjustment for the QE dataset

both BERTScore and METEOR record 20% and 18% higher correlation, respectively. It is also worthy of notice, that although the Pearson correlation of BLEU improves from 0.16 to 0.33 with the SAM adjustment, the correlation is still relatively small. Knowing that BLEU score is usually the gate-keeping tool for accepting improvement in MT research, the results cast doubt on the efficacy of this non-semantic method to penalise sentiment-critical translation errors.

6 Error Analysis and Limitations

Human sentiment can be expressed in intricately subtle ways so that the mistranslation of the affect message is not necessarily reflected in divergence of polarity scores. We conducted an error analysis on instances where the SAM adjustment scores were not able to capture the MT’s failure to transfer the correct sentiment due to different linguistic phenomena. The first phenomenon we identified is related to structure shifting. For example, the sentiment distance between the source tweet ‘I was saddened by him’ and its mistranslation ‘I made him sad’ is very small despite the flipping of sentiment direction in the translation. The three metrics with and without the SAM adjustment failed to penalise this type of distortion in the QE dataset. Second, some words in the lexicon did not have a score representative of their sentiment weight. For example, most prepositions in the SentiWord lexicon are neutral, yet by checking the data, it was found out that a mistranslation of a preposition can distort the affect message. For example, the source tweet ‘What is the benefit of me in this world’ was mistranslated by the MT system as ‘What is the benefit for me in this world’; the wrong preposition causes the translation to lose the sad tone in the source tweet. Again, similar instances were not adequately measured with the three metrics with and without the SAM adjustment. Third, some nuanced sentiment-carrying words specific for the informal style of tweets caused a mistranslation of sentiment which was not captured by our approach. For instance, a one-word tweet referring to a political figure as ‘Prick’ was mistranslated as ‘Sting’. The source is a slang word used to refer to a mean, contemptible man. The translation wrongly received a high score since both ‘prick’ and ‘sting’ had very similar negative sentiment weights. The translation, however, fails to reflect the aggressive sentiment in the source tweet. The BLEU metric, on the other hand, succeeded in giving a penalty to similar short mistranslated tweets without the SAM adjustment.

One other limitation to the current approach for assessing the translation of sentiment is that it relies on an English sentiment lexicon. The applicability of this approach to other languages depends on the availability of a similar high-precision and high-coverage sentiment lexicon. We have overcome this problem by using the English backtranslations of the Arabic and Spanish tweets in the QE dataset. There are, however, multilingual translations of English sentiment lexica that are commonly used in the sentiment-analysis of non-English text. It remains to be tested how far a translated sentiment lexicon is capable of measuring sentiment transfer among different language pairs by the SAM scoring approach.

7 Conclusion

The most frequent scenario in which an MT system is used to translate sentiment-oriented text is for the translation of online UGC such as tweets, reviews or social media posts. The users of the MT online tool take these translations at face-value as there is
no human intervention for accuracy checking. It is important, therefore, to ensure the reliability of the MT system to accurately transfer the author’s affect message before it is used as an online tool. The reliability of an MT system with such big data is commonly measured by automatic metrics. We have shown that conventionally accepted metrics may not always be an optimum solution for assessing the translation of sentiment-oriented UGC. We presented an empirical approach to quantify the notion of sentiment transfer by an MT system and, more concretely, to modify automatic metrics such that its MT ranking comes closer to a human judgement of a poor or good translation of sentiment. Despite limitations to our approach, the SAM adjustment serves as a proxy to the complicated task of manually evaluating the translation of sentiment across different languages.

Acknowledgements

This research work has been supported in part by the TranSent project at the Centre for Translation Studies (CTS) of the University of Surrey.

References


Saif M. Mohammad and Felipe Bravo-Marquez. 2017. WASSA-2017 shared task on emotion intensity. In Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA), Copenhagen, Denmark.


## Appendix A: Examples for Mistranslated Tweets Measured with and without SAM adjustment

<table>
<thead>
<tr>
<th>MT output</th>
<th>Reference</th>
<th>BLEU/METEOR/BERTScore</th>
<th>BLEU/METEOR/BERTScore+SAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>If he had blew himself up in your country, God would forgive him.</td>
<td>If he had blew himself up in your country, may God <strong>not</strong> forgive.¹</td>
<td>0.70/0.83/0.96</td>
<td>0.35/0.41/0.48</td>
</tr>
<tr>
<td>Currently, I am <strong>excited</strong> about the one who criticizes those who are not happy at Eid.</td>
<td>Currently, I am <strong>angry</strong> about the one who criticizes those who are not happy at Eid.</td>
<td>0.77/0.52/0.89</td>
<td>0.23/0.15/0.26</td>
</tr>
<tr>
<td>My study skills are <strong>awesome</strong>.</td>
<td>My study skills are <strong>terrible</strong>.</td>
<td>0.40/0.41/0.85</td>
<td>0.11/0.11/0.27</td>
</tr>
<tr>
<td>May God bless you, the destroyer of Bahrain.</td>
<td>May God <strong>not</strong> bless you, the destroyer of Bahrain.</td>
<td>0.63/0.53/0.60</td>
<td>0.35/0.29/0.33</td>
</tr>
<tr>
<td><strong>My brother</strong>, this match looks like the super match.</td>
<td><strong>I'm afraid</strong> this match will look like the super match.</td>
<td>0.25/0.63/0.90</td>
<td>0.05/0.18/0.22</td>
</tr>
</tbody>
</table>

¹ Translation errors are underlined.
Abstract

There is an incredible amount of information available in the form of textual documents due to the growth of information sources. In order to get the information into an actionable way, it is common to use information extraction and more specifically the event extraction, it became crucial in various domains even in public health. In this paper, We address the problem of the epidemic event extraction in potentially any language, so that we tested different corpora on an existed multilingual system for tele-epidemiology: the Data Analysis for Information Extraction in any Language (DANIEL) system. We focused on the influence of the number of documents on the performance of the system, on average results show that it is able to achieve a precision and recall around 82% but when we resorted to the evaluation by event by checking whether it has been really detected or not, results are not satisfactory according to this paper’s evaluation. Our idea is to propose a system that uses an ontology which includes information in different languages and covers specific epidemiological concepts, it is also based on the multilingual open information extraction for the relation extraction step to reduce the expert intervention and to restrict the content for each text. We describe a methodology of five main stages: Pre-processing, relation extraction, named entity recognition (NER), event recognition and the matching between the information extracted and the ontology.

1 Introduction

Infectious diseases are responsible for the morbidity and mortality and like we see today with the Covid-19 pandemic, surveillance provides us with information to improve our knowledge of their epidemiology (space-time dynamics, evolution of clinical and microbiological characteristics), in order to identify an appropriate control and prevention measures. The growth of digital data sources provide an avenue for data-driven surveillance, referred to as Epidemic Intelligence. The purpose of epidemic intelligence is to detect, analyze and monitor potential health threats over time (Nash and Geng, 2020). It requires the development of tools dedicated to the collection and processing of unstructured textual data published on the Web. Information Extraction (IE) (Martinez-Rodriguez et al., 2020) is one of the areas of active research in artificial intelligence and made it possible to analyze data from web sources. Different tasks of the IE systems have been suggested such as named entity recognition (NER) to identify real-world objects such as names of people, locations, names of diseases, etc. The relation extraction, which aims to find a semantic relation between two entities in a text (Elloumi et al., 2012). The event extraction is also an important task in the field of IE to detect, from the text, the occurrence of events with specific types, and to extract arguments (i.e. typed participants or attributes) that are associated with an event (Hettiarachchi et al., 2021). In 2007, the open information extraction has appeared and has introduced a new extraction paradigm unlike the traditional IE methods, relations are automatically detected instead of specifying target relations in advance and it enables a fast extraction over huge datasets (Niklaus et al., 2018). In this paper, we applied some experiments on the DANIEL system in order to evaluate its performance especially in the multilingual event extraction in the epidemiological field and we have exploited corpora from several diverse language families namely, English, Greek, French, Spanish, Portuguese, Russian, Polish, and Chinese. On the latter we proposed a new approach which reduces the expert intervention by using a multilingual OIE systems for a relation extraction, an automatic NER, and an ontology applied for any epidemiological event. The remainder of this paper...
Figure 1: The architecture of the DANIEL system

is organized as follows. Section 2 reviews works related to epidemic surveillance and event extraction systems, Section 3 describes the dataset used in the experimental study, in Section 4 we present the results. Finally, we provide a discussion about our new proposition and a conclusion in Section 5.

2 Related work

Many systems have been proposed in the domain of IE and which reflects the basis that were taken to build our approach. We made a study on three different approaches that take us to our perspectives for our new system.

2.1 The DANIEL System

The system proposes techniques for identifying emerging infectious disease threats in online news text, it’s based on two main steps: Corpus creation and evaluation. Figure 1 illustrates the architecture of the DANIEL system (Mutuvi et al., 2020a).

**Corpus creation** The system uses the Program for Monitoring Emerging Diseases (PROMED) platform to create the corpus. Firstly, the source URLs where the article was originally published, together with the other meta-data such as title, description, location, date were formatted and stored in a JSON format making the corpus easily reusable and reproducible. Then a language filtering was performed to ensure that only documents belonging to the languages of interest were retained using the K-means clustering algorithm. The boilerplate content such as navigation links, headers and footers was eliminated from HTML pages, it is among the data cleaning tasks. The final pre-processing task was de-duplication which involves eliminating perfect duplicate and near-duplicate content so that only one instance of each text was preserved.

**Evaluation** DANIEL is a multilingual news surveillance system, it aims to extract disease-location for each text in its corresponding language. It describes an event as a disease outbreak and the place where it occurred. It avoids grammar analysis and the usage of language-specific NLP toolkits (e.g., Part-of-speech tagger, dependency parser), it considers the text as a sequence of strings instead of words. The named entities presented by a list of diseases and a list of locations in JSON files in different languages. The named entity extraction depends on a ratio \( r \) which has a default value that can be fine-tuned by the user. The ratio \( r \) is depicted in the following equation:

\[
\frac{\text{length}(\text{substring})}{\text{length}(\text{entity name})}
\]

This ratio is a kind of threshold for the different size of the substrings. For example \((r=0.8)\) means that substrings sharing 80% of the named entity.

The news articles are grouped into various categories such as politics, wellness, travel, entertainment, sports and healthy living, among others. The models classify a news article as either relevant or non-relevant, depending on whether it alerts about a disease outbreak or not as described in (Mutuvi et al., 2020a).

2.2 The BioCaster System

BioCaster (Collier et al., 2008) ingests documents through RSS feeds. An Automatic classification of the reports was performed for topical relevance using a naïve Bayes algorithm. A named entity recognition is then accomplished for relevant documents for 18 term types based on the BioCaster ontology. The BioCaster ontology includes information in eight languages focused on the epidemiological role of pathogens as well as geographical locations with their latitudes/longitudes. At this stage disease-location pairs are plotted onto a public portal called the Global Health Monitor, to gain a geographically contextualized view of an outbreak anywhere in the world in Google Maps which can be filtered by pathogen, syndrome or text type. The event extraction is based on matching entity classes, skipwords, string literals, regular expressions, entity types as well as guard lists which include verbs of infection, common victim expressions. This is done by using a Simple Rule Language (SRL) as described in (Collier et al., 2008).
2.3 Event Detection based on Open Information Extraction and Ontology System

The approach proposes an event extraction by using an OIE system for a relation extraction without supervision, an automatic NER, and an ontology applied for any management change event (Sahnoun et al., 2020). The approach admits 2 phases as shown in Figure 2 that depend on each other: A learning phase and a recognition phase. The learning phase consists of modeling an event by an ontology (classes, subclasses and instances in relations), and constructing a set of rules manually. The recognition phase includes the RE, the NER and an automatic reasoning between the rules and the input ontology adaptation.

The rules construction is an important step which drives to a possible event extraction. For the relation extraction the system uses OLLIE, it is an Open information extraction tool which extracts the relationship triplets that contains three textual components (Arg1, Rel, Arg2) where the first and the third stand for the pair of arguments and the second indicates the relationship. For the Named Entity Recognition step, the system can detect a person, an organization, location, etc., in any part of the triplet using the python library SPACY1.

For the ontology input adaptation, verbs will be passed through a lemmatization layer to convert verbs to their infinitive form. An instance is every token recognized by a NE, then it will be added to the ontology and linked by relations whenever the following conditions are achieved:

- The number of named entities is greater than or equal to 2 to have a possible relationship among them.
- The lemmatized verb and the other relations between delimiters “;” should be included in the relation list of the ontology and Named Entities can be linked with these relations.

The last step is reasoning by inferring logical consequences of a set of rules to affect for each instance its role (event) as described in (Sahnoun et al., 2020).

3 Dataset and Evaluation

We tested the Daniel system on three large datasets in different languages (low- or high-resource), the first (Daniel-corpus) contains 2089 of relevant and irrelevant files (Romain et al., 2013), then we extend the dataset (BIG-corpus) (Mutuvi et al., 2020b) to include additional languages so that it covers news articles from several, diverse language families: Germanic (English, en), Hellenic (Greek, el), Romance (French, fr), Slavic (Polish, pl and Russian, ru) and Sino-tibetan (Chinese, zh). It includes 7046 irrelevant files and 1653 relevant files, and the third present the fusion of the two precedent corpora. The statistics of the dataset are presented in Table 1 and Table 2. The experimental study was carried out according to the two measurements of the true positive rate (TPR), the false positive rate (FPR), depending on the threshold r (1). The true positive rate (TPR) (2): Called also sensibility measures the likelihood of actual positive results. The false positive rate (FPR) (3): It’s the probability that a positive result will be given when the true value is negative.

\[
TPR = \frac{TP}{TP + FN} = \text{Recall} \quad (2)
\]

\[
FPR = \frac{FP}{FP + TN} \quad (3)
\]

We used a ROC curve to visualize the performance of the binary classifier. It’s a plot of the TPR versus the FPR for every possible classification threshold. A classifier that does a very good job separating the classes will have a ROC that hugs the upper left corner of the plot 2. Conversely, a classifier that does a poor job will have a ROC.

1https://www.ekino.com/articles/handson-de-quelques-taches-courantes-en-nlp

2https://www.dataschool.io/roc-curves-and-auc-explained/
curve that is close to the diagonal line. The purpose of AUC, which stands for Area Under the Curve. A very poor classifier has an AUC of around 0.5 and a perfect classifier has an AUC close to 1. After Evaluating the Daniel system and by taking into account the influence of the number of documents on the performance of the system, we resorted to the evaluation by event, each disease-location pair (e.g. flu in Spain) is considered as a unique event, regardless of the number of documents in which it has been reported, and then we check whether it has been detected or not.

Figure 3: ROC curve for the DANIEL corpus (2089 files) (Romain et al., 2013)

### 4 Results

#### 4.1 Evaluation By Documents

Figure 3 depicts the tracing of the ROC curve for the first corpus (Daniel-corpus) for 2089 files, the system represents its performance with a threshold of 0.8: The TPR has for percentage of 89% while we find only 4% of FPR that explains that the system has detected a significant number of events in the relevant documents. The Daniel-corpus contains texts in five different languages: Greek, English, Chinese, Russian and Polish as it shows in Figure 5. For the five languages, the ROC curve hugs the upper left corner of the plot for each curve (i.e. the system detects well the events for each language). Figure 4 demonstrate that the ROC curve is close to the diagonal for the BIG-corpus, we have 39% of TPR and 10% of FPR for a threshold of 0.8 results are not satisfactory for this corpus, so when we have increased the number of documents the TPR decreased. We merged the two corpora in order to obtain a more representative corpus, the results of the subsequent corpus are shown in figures 6 and 7. The results have been slightly improved for Spanish, Portuguese, Greek, Russian, Polish and Chinese the system gives a high-performance but this not the case for English and French. For a threshold of 0.8 we observe a TPR of 42% and a 9% FPR.

#### 4.2 Evaluation By Event

Table 3 shows the results of the evaluation by event for the DANIEL-corpus, demonstrating that there are 66 events out of 86 are missed if no entity linking is performed. The total number of unique
events in the corpus is not the sum of unique events in each subcorpus. In the cumulated corpora there are 16 events that are reported in more than one language.

<table>
<thead>
<tr>
<th>Language</th>
<th># Events</th>
<th>Detected</th>
<th>Missed</th>
<th>Detected %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chinese</td>
<td>9</td>
<td>1</td>
<td>8</td>
<td>89%</td>
</tr>
<tr>
<td>English</td>
<td>19</td>
<td>6</td>
<td>13</td>
<td>68%</td>
</tr>
<tr>
<td>Greek</td>
<td>16</td>
<td>5</td>
<td>11</td>
<td>69%</td>
</tr>
<tr>
<td>Polish</td>
<td>21</td>
<td>4</td>
<td>17</td>
<td>81%</td>
</tr>
<tr>
<td>Russian</td>
<td>21</td>
<td>4</td>
<td>17</td>
<td>81%</td>
</tr>
<tr>
<td>All</td>
<td>70</td>
<td>18</td>
<td>52</td>
<td>74%</td>
</tr>
</tbody>
</table>

Table 3: Evaluation by unique event for the DANIEL-corpus with a threshold of 0.8 (NB: this a strict scenario where no entity linking has been performed)

The results of the relevant documents for The BIG-corpus in Table 4 shows that there is 75 of events detected between 939 unique events (in all languages) so the number of events detected is not great if we consider that we have a much larger corpus. Table 5 presents an example of an event extraction in the form of a peer of disease-location, the real date that documents have mentioned the event and the date when it was detected.

5 Conclusion and Perspectives

In this paper we have focused on the study of some systems in the epidemiological field such as Daniel and BioCaster, and an event extraction system which is based on a methodology that opens the door for a new proposition using the open information extraction and the ontology. We propose a procedure to an eventual epidemic event extraction
Table 4: Evaluation by unique event for the BIG corpus for a threshold of 0.8 (NB: this is a strict scenario where no entity linking has been performed)

<table>
<thead>
<tr>
<th>Language</th>
<th># Events</th>
<th>Detected</th>
<th>Missed</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>328</td>
<td>20</td>
<td>308</td>
<td>93%</td>
</tr>
<tr>
<td>French</td>
<td>169</td>
<td>20</td>
<td>149</td>
<td>88%</td>
</tr>
<tr>
<td>Spanish</td>
<td>278</td>
<td>33</td>
<td>245</td>
<td>88%</td>
</tr>
<tr>
<td>Portuguese</td>
<td>183</td>
<td>2</td>
<td>181</td>
<td>99%</td>
</tr>
<tr>
<td>All</td>
<td>939</td>
<td>75</td>
<td>864</td>
<td>92%</td>
</tr>
</tbody>
</table>

Table 5: Example of an Event Extraction

<table>
<thead>
<tr>
<th>Event</th>
<th>Real date</th>
<th>Detection date</th>
<th>Diff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flu-India</td>
<td>12-01-2012</td>
<td>17-01-2010</td>
<td>5 days</td>
</tr>
<tr>
<td>Grypa-Chinu</td>
<td>06-01-2012</td>
<td>12-01-2012</td>
<td>6 days</td>
</tr>
<tr>
<td>H5N1-Kiwu</td>
<td>30-12-2011</td>
<td>30-12-2011</td>
<td>0 days</td>
</tr>
</tbody>
</table>

which consists of five main stages: Pre-processing, relation extraction, named entity extraction and the matching between the information extracted and the ontology. The first task is to Retrieve articles from PROMED in different languages, then a pre-processing step based on a Data cleaning task to Eliminate the boilerplate content from the corpus.

The text then will be passed to a relation extraction using an open information extraction to restrict the content of a text into triplets of relationships so that it is in form (Arg1, verb, Arg2) and we can differentiate between the nominal part and the verbal part. In the nominal part where we can find the named entities and in the verbal part we can visualize verbal expressions in the epidemiological context. The relation extraction method aim to represent semantic relations between entities. The entities have numerous applications in building knowledge representation models that report relations between words, such as ontologies, semantic networks.. In this work, we will investigate the area of multilingual open information extraction for the Portuguese, English and other languages (Claro et al., 2019). The extracted relations will be passed to a named entity recognition layer. The system can detect the location, the date, the percentage and the name of disease... For the event extraction we can use the simple rule language (SRL) with a capability to match regular expressions and guard lists include verbs of infection, common victim expressions, occupation names. The ontology in our approach present the event which is an object that admits an existence in the space of time and depends on other objects in relation. An ontology is a set of concepts, as well as relationships between these concepts that’s why the event was modeled by an ontology. The matching between the extracted information and the ontology will bring us to an eventual event extraction as depicted in Figure. 8, we can use an ontology already existed like the BioCaster ontology but it is not available online so we’re going to do it ourselves.

We are looking for evaluating the results of the test obtained and compare them by another systems like BioCaster and DANIEL.

References


Exploiting Domain-Specific Knowledge for Judgment Prediction is no Panacea

Olivier Salaün¹, Philippe Langlais¹, and Karim Benyekhlef²

¹RALI, DIRO, University of Montréal
{salaunol,felipe}@iro.umontreal.ca
²Cyberjustice Laboratory, Faculty of Law, University of Montréal
karim.benyekhlef@umontreal.ca

Abstract

Legal judgment prediction (LJP) usually consists in a text classification task aimed at predicting the verdict on the basis of the fact description. The literature shows that the use of articles as input features helps improve the classification performance. In this work, we designed a verdict prediction task based on landlord-tenant disputes and we applied BERT-based models to which we fed different article-based features. Although the results obtained are consistent with the literature, the improvements with the articles are mostly obtained with the most frequent labels, suggesting that pre-trained and fine-tuned transformer-based models are not scalable as is for legal reasoning in real life scenarios as they would only excel in accurately predicting the most recurrent verdicts to the detriment of other legal outcomes.

1 Introduction

At the intersection of machine learning and law, legal judgment prediction (LJP) is a task that consists in predicting either the outcome of a lawsuit (Skalak, 1989; Nallapati and Manning, 2008; Katz et al., 2017; Aletras et al., 2016; Liu and Chen, 2017) or some other case attributes such as legal areas (Șulea et al., 2017; Soh et al., 2019) or charges (Xiao et al., 2018).

One specificity of court rulings is that they are based on the application of legal articles to the factual description of the case. That is to say, a judge must determine whether some law articles are relevant to a case, and if applicable, whether the legal principles they embody are violated. Therefore, articles as domain-specific knowledge can be used as leverage for improving LJP performance, as shown by Luo et al. (2017) and Long et al. (2019) for charge prediction and divorce judgment prediction respectively. Xu et al. (2020) also went further by using articles for distinguishing confusing charges in a charge prediction task.

Meanwhile, transformers (Vaswani et al., 2017) and BERT models (Devlin et al., 2019) in particular have been widely used in NLP tasks with the assumption that such models, if first pre-trained on massive corpora and then fine-tuned on the dataset of a given task, could suffice for achieving significant improvements. On one hand, this turned out to be true with the CAIL2018 dataset (charge prediction task) as shown by Wang et al. (2020). On the other hand, Holzenberger et al. (2020) mentioned in a statutory reasoning entailment task that a transformer model does worse than a rule-based model, even after further pre-training on the domain corpus. Furthermore, in an employment notice prediction task, Lam et al. (2020) emphasized that domain adaption of such models could even harm performance. These elements raise the question of how well a pre-trained transformer model can handle a legal NLP task and how well the input from domain-specific knowledge such as legislative text can improve the LJP performance. To the best of our knowledge, in the case of LJP tasks aimed at verdict prediction, no experiment has tested so far the application of pre-trained BERT models on both tribunal decision text and cited law articles text combined altogether, which we intend to do in this work.

We designed a multilabel classification task in which the model must predict the ruling outcomes on the basis of the facts description. One can imagine that such a predictive engine could be used for legal assistance for those who may not afford the services of a legal expert. Unlike Luo et al. (2017), we put the article prediction aside in order to focus solely on the verdict prediction and assess in which conditions input article-based features can improve classification. For our experiments, we use a landlord-tenant disputes corpus used by West-
erman et al. (2019) and Salaün et al. (2020) from which we extracted fine-grained targets labels and article features in order to encompass as much as possible the variety of rulings, thus making the task more representative of real life cases. We present the preprocessing of the dataset along with the creation of article-based features in Section 2. The architectures of the models are shown in Section 3 along with three methods for integrating the information from the articles mentioned in the decisions. Discussion of the results and concluding remarks are provided in Sections 4 and 5 respectively.

2 Preparation of the Dataset

The Administrative Housing Tribunal is a court of Quebec in Canada with an exclusive jurisdiction in landlord-tenant disputes. We got access to an exhaustive corpus of 667,305 decisions in French issued from 2001 to 2018 publicly accessible through SOQUIJ portal. Documents average and median lengths amount to 307 and 235 tokens respectively with a standard deviation of 371.

Figure 1: Excerpt of a decision translated from French. The text in italics is the verdict while underlined text contains references to articles.

Each decision is split in two by applying heuristics based on the syntax of the documents: the pre-verdict section, used as text input (text before the italics in Figure 1), and the verdict section containing the legal solution chosen by the judge in charge of the case (text in italics in Figure 1). The pre-processing of both sections are described in subsections 2.1 and 2.2 respectively.

2.1 Preprocessing of Input Features

As one of our goal is to assess the conditions in which articles help to improve predictions, we applied heuristics on the pre-verdict text to extract a total of 1,790 unique cited law articles, 33.8% of which were mentioned only once across the entire corpus. Also, not all articles are related to housing law. We address this by keeping only 445 articles from the Book Five - Obligations of Civil Code of Quebec (C.C.Q.) which establishes the rules concerning the contractual obligations between landlords and tenants and whose frequency in the corpus has a minimum of two. Three examples are shown on Table 1 for the decision in Figure 1. Article distribution is heavily skewed: the 3 most frequent articles cover 72%, 42% and 27% of all documents respectively while all other articles do not exceed 4%. Mean and median frequency of the articles amount to 2571 and 17 respectively. Section 3 further describes the use of these articles as input.

The pre-verdict section contains both fact description and legal analysis. As the latter can give hints about the verdict that the model is expected to predict, we removed from the pre-verdict section any paragraphs containing citations of articles (underlined text in Figure 1) and we capped the maximum input text length at 128 tokens. By doing so, we force the model to make predictions on the sole basis of fact descriptions.

2.2 Making Target Labels from the Verdict Section of the Decisions

We carefully combined NLP-engineered tools (regular expressions and the like) and some housing law expertise in order to pseudo-automatically identify 23 labels that we believe are representative of the rulings and that cover the diversity of the verdicts at a fine grain. These labels are cumulative and three are shown as an example in Table 1 for the decision in Figure 1. The average and median numbers of labels per decision both amount to 3 with a standard deviation of 1.5. Nearly half of all rulings involves an eviction (48.1%) and a termination of the lease (46.1%), hinting that a significant part of the cases

<table>
<thead>
<tr>
<th>Articles</th>
<th>Target labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1619</td>
<td>monetary penalty for defendant</td>
</tr>
<tr>
<td>1971</td>
<td>eviction</td>
</tr>
<tr>
<td>1883</td>
<td>termination of the lease</td>
</tr>
</tbody>
</table>

Table 1: Civil Code of Quebec articles and verdict labels extracted from the decision shown in Figure 1.
involve an unfavourable outcome for tenants sued by landlords. Further investigation confirms a bias favourable for landlords as 80.3% of cases with the top frequent label `monetary_penalty_for_defendant` have a tenant as the (penalized) defendant.

Overall, 0.05% of all instances were not assigned any labels and 18.2% did not contain any articles. For the design of our experiments, all instances with no article or no verdict label were excluded, thus resulting in a final corpus of 544,857 documents with an average of 3.3 labels per instance (standard deviation of 1.4 and median of 4). The average and median numbers of articles per document both amount to 2. Our instances are randomly divided into training, validation and test sets with a 60-20-20 ratio.

3 Models

We aim at designing a multilabel classification task in which a model has to return the labels corresponding to the verdict on the basis of the pre-verdict section of each decision. Our baseline is a One-Versus-Rest Logistic Regression, i.e. each label has its own classifier. The input text is vectorized through character-based TF-IDF spanning bi-grams to 8-grams (character-based features outperformed token-based). Only the top 100k n-grams are retained. We also use CamemBERT (Martin et al., 2019), a variant of BERT (Devlin et al., 2019) that was pre-trained on generic French corpora. We further pre-trained the `camembert-base` default parameters (unsupervised masked language modelling task) from Wolf et al. (2020) on the train set during 20 epochs. We eventually fine-tuned them during the multilabel classification task. The batch size and the maximum number of fine-tuning epochs are set at 50 and 20 respectively. Training is stopped when no further improvement is obtained in terms of exact match on validation set. We use the Adam optimizer (Kingma and Ba, 2015) with a learning rate at $10^{-5}$ (all other hyperparameters are left at default value). The optimization criterion is the binary cross entropy with logits loss for numerical stability during optimization. Therefore the final output is that of a logit function, with scores ranging from $-\infty$ to $+\infty$. A label is returned whenever its associated output value exceeds 0. We use a vanilla CamemBERT model whose only input is the pre-verdict text and three other variants described in the next subsections.

3.1 One-Hot and Node2Vec Encoding of Articles

For each instance, the mention/absence of each article is one-hot encoded through a 445-dimensional vector, each dimension corresponding to one article. We have one model named BERT-OH (Figure 2 part a) in which the BERT output of a decision text (768-dimensional vector corresponding to the first token from hidden states) and the articles one-hot vector are concatenated and passed through fully connected layers before outputting the verdict labels. Given the heavily skewed distribution of these articles among the documents, these discrete one-hot vectors are sparse and likely not very expressive (in the case of Figure 1 for instance, all dimensions except three would be zeroed because of only three articles extracted as shown in Table 1).

![Architecture diagrams of BERT-OH, BERT-N2V both on part a) and BAF on part b).](https://example.com/figure2.png)

Figure 2: Architecture diagrams of BERT-OH, BERT-N2V both on part a) and BAF on part b).

As a result, we thought of a continuous and more expressive representation that could embed the articles organization within the law. In the one-hot vector approach, each article is assigned one dimension independent from the other as if all articles were completely unrelated to each other. But when having a closer look at the C.C.Q. Book Five - Obligations on Figure 3 that concentrates the rules related to landlord-tenant disputes, the articles are

---

\(^{2}\) Exact match achieved by further pre-trained models is around one percent point greater than models with default pre-trained parameters.

\(^{3}\) Screenshot from CANLII [http://canlii.ca/t/533nd](http://canlii.ca/t/533nd) last accessed on January 20th 2021
actually organized into titles, chapters, divisions, paragraphs and so on, down to the articles themselves. As each subcategory becomes more and more precise, the articles encompassed in it are related to closer and closer legal concepts. For instance, articles 1957 to 1970 are especially dedicated to repossession of a dwelling and eviction and can be expected to relate to the same legal objects. Therefore, we wanted to make an embedding that could capture the structural closeness between articles, that is, two articles located in the same subsection would have closer representations.

Another argument in favour of using embeddings based on the topological relatedness among articles is the fact that articles with close numbers have a tendency to co-occur together in the decisions, as shown on the diagonal of Figure 4 along which distinct articles with close numbers tend to belong to the same subsections and to have higher correlation values.

One method for representing the topological organization of the law is Node2Vec (Grover and Leskovec, 2016). We first built a tree whose root is Book Five - Obligations and added all of the subsequent sections as nodes. Articles were added as leaves. Next, the edges were placed by linking each node/section to the subsequent nodes/sections that it encapsulates. For instance, in Figure 3, CHAPTER IV - LEASE is linked to DIVISION IV - SPECIAL RULES FOR LEASES OF DWELLINGS which is linked to § 7 - Right to maintain occupancy, and so on, until placing the edges between I. - Beneficiaries of the right and each of the leaves/articles 1936 to 1940. An edge between two nodes cannot “skip” any node of intermediate level between the two if there is any. Overall, we made a graph with 1,565 nodes (subsections and articles) and 1,564 edges. From this graph, we made an embedding for each article such that articles belonging to the same subsection and related to the same legal concepts, would have embeddings close to each other. Following the Node2Vec technique, we generated 200 random walks from each node (article or section) that gave a total of 313k sequences of 200 nodes by following the edges. Then, we trained a Word2Vec model (Mikolov et al., 2013) on these sequences during 10 epochs and with a window size of 10, so that each node is assigned a Node2Vec embedding (256-sized vector) that captures the proximity of the neighbouring nodes. We eventually only kept the Node2Vec representations of the leaves that correspond to the 445 retained articles. For the sake of clarity, the vectors of two articles randomly drawn would have a higher cosine similarity if the articles belong to the same subsection than if they belong to distinct ones. For the BERT-N2V (node2vec) model, the input contains the pre-verdict text (passed through CamemBERT) plus an average of the Node2Vec embeddings of all articles mentioned in the documents (e.g. average embedding of articles 1619, 1971 and 1883 in Table 1 for the case in Figure 1).

3.2 Applying BERT on the Law Articles Text

The one-hot and the Node2Vec encodings are still shallow representations of the articles as they do not even use the text of Civil Code of Quebec. At most, Node2Vec is only capturing a topological representation of how the 445 articles are organized.
in the law. This is why we considered another model called BAFA (BERT model with Adapters applied on Facts and Articles, Figure 2 part b) that is given as input the pre-verdict text of the decision and the text of all articles cited in it. The 445 retained articles have a average length of 34 tokens (median of 32 and standard deviation of 17). The pre-verdict section and the articles are encoded through two distinct CamemBERT models so that one is fine-tuned on the decision text and the other on the law text. The BERT output (i.e. first token of hidden states) of the decision is then passed through a 12-head attention mechanism as a query while the BERT output obtained from each cited articles are concatenated (up to 22 cited articles per decision) and passed as a key-value pair. The output of the attention mechanism is then passed through two fully connected layers. As we use two distinct CamemBERT modules, the batch size is reduced to 4 and we added adapters (Pfeiffer et al., 2020) as an attempt to speed up computation.

To the best of our knowledge, there is no other work that uses the fine-tuning of pre-trained BERT models on the text of cited articles for verdict prediction of court decisions. When it comes to LJP tasks formalized as text classification, many of most recent works usually aimed at charge prediction or law articles prediction on the sole basis of the facts description (Xiao et al., 2018; Zhong et al., 2018). Şulea et al. (2017) made a ruling prediction task comparable to ours but without the text of the articles. When it comes to experiments that actually use the text of law articles, Hu et al. (2018) and Xu et al. (2020) used it for improving prediction of confusing charges only while Luo et al. (2017) and Long et al. (2019) used a combination of recurrent neural nets with attention mechanisms for encoding it into their models for charge prediction and divorce verdict respectively. Still, none of these works involve transformer architecture. Concerning the experiments that use BERT, Chalkidis et al. (2019) and Wang et al. (2020) used pre-trained models for prediction of violation of human rights article and of charges respectively on the basis of the facts only.

4 Results and Discussion

The classification results are shown in Table 2. For each label, we compute the F1 score (harmonic mean between precision and recall) obtained by each model and add the label distribution across the test documents. For each model, the last two rows of Table 2 present two overall scores based on metrics that we believe are constraining enough and appropriate for the evaluation of systems that could one day be deployed in real life scenarios. F1 macro average is unweighted average of all labels F1 scores, and thus penalizes models that delivers poor F1 scores for a large number of labels. It measures the ability of the model to predict a large variety of rulings. We also compute exact match which corresponds to the ratio of instances for which a model is able to return the exact set of labels assigned to them. Therefore, an instance is considered as misclassified whenever its prediction has a label in excess or one missing.

4.1 Gains Obtained across Verdict Labels with Article-Based Features

One goal of our experiment is to assess how articles can improve the prediction of cases in which they are cited. Figure 5 shows a heatmap detailing the correlation among articles and verdict labels. In the top left corner, monetary_penalty_for_defendant, eviction, termination_lease and proportional_enforcement are strongly correlated with articles 1619, 1971 and 1883 which respectively define: the computation of an additional indemnity that can be added to damages; a rule that allows the termination of the lease if rent is over three weeks late in payment; a rule so that the tenant may avoid lease termination by paying the due rent plus interest before the judgment. Although these articles make a consistent legal ground with the aforementioned verdict labels, inputting them into the models through any representation (be it one-hot/node2vec/BERT encoding) added very little improvement for the F1 scores of these labels (by at most 2.8 percent point on average relative to CamemBERT alone), very likely because of their already high frequency in the corpus. Furthermore, on the heatmap on Figure 5, tenant_ordered_pay_rent is strongly correlated with article 1973 that defines the conditions allowing the judge to grant lease termination (unless the payment of the rent is over three weeks late, the judge may choose to either terminate the lease immediately or either order the tenant to pay the rent) and the article-based features help in dramatically improving the corresponding F1 score by 18-22 percent points compared to a sole CamemBERT setting. We also observe that landlord_repossesses_rental_unit and
monetary_penal­ty_for_applicant are strongly asso­ciated with several articles, especially 1963 and 1967 which establish respectively the conditions by which the judge can authorize a landlord-applicant to repossess their dwelling from which a tenant refuses to depart and the indemnities that the landlord must pay to the tenant for moving expenses when repossession is granted. Article-based models also improved the F1 scores of these two labels, though not as important as for tenant_ordered_pay_rent, with average gains of 4.1 and 10.1 points respectively. All in all, the inclusion of article-based features has a negligible impact when the labels already have a high support in the documents, but the improvement is more significant for labels that are rarer (landlord_repossesses_rental_unit, monetary_penalty_for_applicant and tenant_ordered_pay_rent have supports below 5%) and that have a high correlation value with the articles cited in them (for the three aforementioned labels and their articles, the average correlation is around 0.75). A counter-example to that principle would be agreement_between_parties and tribunal_sets_new_rent whose correlation values with cited articles are not that important (below 0.5) and for which no significant improvement on F1 scores is observed with article-based features.

### 4.2 Comparing Performance among Models

In general, BERT-based models do better than the baseline in terms of macro-averaged F1 score and exact match. Furthermore, the scores show that article-based features help in outperforming a sole CamemBERT model with higher exact match scores by up to 3.1 percent points and a higher F1 score macro-average by up to 3.8 points. The best F1 macro average score is achieved by the model with node2vec (63.2%) while best exact match score is obtained by CamemBERT with one-hot vectors (67.0%). Still, such results must be nuanced: the performance gains are mostly obtained with either high frequency labels (five most frequent plus lease_already_terminated) or labels that are strongly correlated with certain articles, which can explain the marginal improvements achieved in the coarse scores at the bottom of Table 2. Furthermore, the use of article-based features...
Figure 5: Heatmap of the correlation matrix of the verdict labels and the 30 most frequent articles. The verdict labels and the articles are sorted by decreasing frequency on their respective axes.

seem to sometimes harm the performance for low-frequency labels relative to vanilla BERT (defendant ordered some action, penalty misc, tribunal cancels past ruling, discontinuance claim, schedule new audience, tribunal upholds past ruling), which suggests that such features add noise rather than help the model in accurately predicting these verdicts.

Although BAFA used the text of both decisions and articles, both encoded through two distinct BERT models, the overall performance is disappointing compared to the variants with one-hot and node2vec features: despite achieving the best F1 scores for some top frequent labels, the overall coarse scores remain below those of BERT-OH and BERT-N2V. Plus, the training of BAFA also is significantly longer4.

For illustrative purposes, if we were to compare the performance achieved by our models in this task with other similar works, we could cite:

- Aletras et al. (2016) and Chalkidis et al. (2019) who achieved respectively an accuracy of 79% and a F1 macro-average score of 80.2% in binary classifications for violation of human rights;

- Şulea et al. (2017) who achieved an accuracy of 92.8% for an 8-mutually-exclusive classes classification task for ruling prediction (our task has 23 cumulative labels);

- Luo et al. (2017) got a macro-average F1 score of 95.4% in a charge prediction task that use articles text as input; in another charge prediction task, Zhong et al. (2018) achieved 49.1% and 70.9% for that score on two other datasets (our task is about verdict prediction).

Two main points can be made from these results. First of all, shallow articles embeddings (one-hot and node2vec) do better than BERT-encoding of law text at allowing a marginal improvement over some low-support labels (though not all), with BERT-N2V reaching the highest F1 macro-average score at 63.2%. A tentative explanation is that directly inputting the text of the cited articles adds noisy information that confuses rather than helps the model in the task while a “fuzzier” representation of the articles gives a broader information about articles (Node2Vec embeds the topological position of articles in Civil Code of Quebec) without forcing the model to combine the legal terminology of the law and the text input of the decisions. The second point is that although article-based models outperform vanilla CamemBERT, this is mainly due to the marginal improvement over some of the top frequent labels and to the improvement over some verdict that are strongly correlated with certain articles. This suggests that such models only excel in predicting the most recurrent and stereotypical landlord-tenant disputes (eviction of a non-paying tenant; moving indemnities for a tenant whose rental unit is repossessed by the landlord).

All in all, these models would be unusable for providing legal assistance for a large variety of cases as they would only excel in predicting accurately the most frequent rulings to the detriment of other types of cases. Also, as these models deal with housing law domain that is related to sensitive social issues (Gallié et al. (2016) emphasized that tenants are less confident in dealing with judicial proceedings), we tried to extract some significant patterns from the self-attention weights in the CamemBERT architecture that could help in understanding what causes the model to return some prediction, but found nothing prone to interpretation, which is consistent with a statement from Jain and Wallace (2019).

4.3 Discussion about the Experiment Setting

The results obtained seem consistent with observations from Holzenberger et al. (2020) who stated that even a further pre-trained BERT model struggles with a legal entailment task, thus suggesting that the fine-tuning of pre-trained BERT models on statutes and law articles text is not sufficient

4Over 1 hour per epoch for a one-BERT model, over 14 hours per epoch for two BERTs combined.
for solving tasks in very specific domains such as tax law or housing law. Regardless of the method used for inputting articles into the models, all of the approaches combining description of the facts and articles just excel at predicting the most frequent verdicts, which suggests that they would be unusable as is at a higher scale as they would not be able to provide satisfactory legal assistance for cases different from the most recurrent ones. Bender et al. (2021) emphasized the risks involved in using large pre-trained models that tend to encode and amplify biases already present in the training data. To paraphrase the title of their paper, their remarks are consistent with our observations as we end up with “legal parrots” which would not be able to accurately address the variety of real world landlord-tenant disputes.

The fact that article-based features allow for significant improvement under the condition that the labels are strongly correlated with articles also raises questions about the setting of LJP experiments: in the charge prediction task made by Xiao et al. (2018), charges and laws with frequency below 30 were removed from the CAIL2018 dataset and each charge label is strongly associated with one specific article. In contrast, in our corpus, each label is not always strongly correlated with some law article, as shown in Figure 5. Some works using the CAIL2018 dataset such as that from Wang et al. (2020) made further changes in the dataset by removing target labels with frequency below 100. Unlike them, in our work, we were much more permissive during the creation of our corpus as we retained articles with a frequency of at least 2 and made labels to exhaustively cover as many verdicts as possible (i.e. 1102 unique combinations of labels), even though some labels could have been merged together (e.g. landlord_repossesses_rental_unit and monetary_penalty_for_applicant tend to co-occur together) or discarded/weighted down. For instance, schedule_new_audience and applicant_forbidden_seek_recourse have a low frequency and are rather technical legal details that would be more relevant for a legal expert than for a layman seeking general advice. If we computed F1 average score weighted by each label’s support, BERT models would have an average performance of 91.6%, but that coarse metric is mostly pulled upwards by scores achieved for most frequent labels. We must also emphasize that in our dataset there is no 1-to-1 correspondence between labels and articles as in CAIL2018 in which articles not relevant to specific charges were removed beforehand. This illustrates the difficulty in automating legal reasoning over cases and unfiltered law articles in a realistic context.

5 Concluding Remarks

We designed a LJP task as text multilabel classification for verdict prediction based on a collection of landlord-tenant disputes in French for which we used a further pre-trained CamemBERT model and applied different types of features derived from the articles cited in the decisions (one-hot, Node2Vec, BERT encoding of the text of articles). By doing so, we noticed that leveraging articles as input features (regardless of the representation used) made either marginal improvements for F1 scores of most frequent labels, either significant improvements for labels that are strongly correlated with certain articles. The use of article-based one-hot features achieves best exact match score (67.0%) while node2vec features achieve best F1 macro average score (63.2%). The model that encodes the text of the articles with BERT does not outperform the two previous methods.

As future work, we plan on comparing how models perform under both “realistic” setting (several rare target labels with no or few connections with the law available, as we did in this work) and “laboratory” setting (where low frequency targets and laws are aggressively filtered out). We also plan to assess whether the patterns observed in our work (performance improves when articles are strongly correlated with labels) also exist in other LJP datasets beyond housing law and Canadian cases. Furthermore, we plan to study further the attention weights and the mechanisms underlying the significant prediction improvement observed for certain labels when the input contains the text of articles that are highly correlated with the corresponding verdicts.

Acknowledgements

We would like to thank the Cyberjustice Laboratory at the Université de Montréal, the LexUM Chair on Legal Information and the Autonomy through Cyberjustice Technologies project for supporting this research.
References


Masking and Transformer-based Models for Hyperpartisanship Detection in News

Javier Sánchez-Junquera
Universitat Politècnica de València
jjsjunquera@gmail.com

Paolo Rosso
Universitat Politècnica de València
prosso@dsic.upv.es

Manuel Montes-y-Gómez
Instituto Nacional de Astrofísica, Óptica y Electrónica
mmontesg@inaoep.mx

Simone P. Ponzetto
University of Mannheim
simone@informatik.uni-mannheim.de

Abstract
Hyperpartisan news show an extreme manipulation of reality based on an underlying and extreme ideological orientation. Because of its harmful effects at reinforcing one’s bias and the posterior behavior of people, hyperpartisan news detection has become an important task for computational linguists. In this paper, we evaluate two different approaches to detect hyperpartisan news. First, a text masking technique that allows us to compare style vs. topic-related features in a different perspective from previous work. Second, the transformer-based models BERT, XLM-RoBERTa, and M-BERT, known for their ability to capture semantic and syntactic patterns in the same representation. Our results corroborate previous research on this task in that topic-related features yield better results than style-based ones, although they also highlight the relevance of using higher-length n-grams. Furthermore, they show that transformer-based models are more effective than traditional methods, but this at the cost of greater computational complexity and lack of transparency. Based on our experiments, we conclude that the beginning of the news show relevant information for the transformers at distinguishing effectively between left-wing, mainstream, and right-wing orientations.

1 Introduction
Media such as radio, TV channels, and newspapers control which information spreads and how it does it. The aim is often not only to inform readers but also to influence public opinion on specific topics from a hyperpartisan perspective.

Social media, in particular, have become the default channel for many people to access information and express ideas and opinions. The most relevant and positive effect is the democratization of information and knowledge but there are also undesired effects. One of them is that social media foster information bubbles: every user may end up receiving only the information that matches his/her personal biases, beliefs, tastes and points of view. Because of this, social media are a breeding ground for the propagation of fake news: when a piece of news outrages us or matches our beliefs, we tend to share it without checking its veracity; and, on the other hand, content selection algorithms in social media give credit to this type of popularity because of the click-based economy on which their business are based. Another harmful effect is that the relative anonymity of social networks facilitates the propagation of toxic, hate and exclusion messages. Therefore, social media contribute to the misinformation and polarization of society, as we have recently witnessed in the last presidential elections in USA or the Brexit referendum. Clearly, the polarization of society and its underlying discourses are not limited to social media, but rather reflected also in political dynamics (e.g., like those found in the US Congress (Andris et al., 2015)): even in this domain, however, social media can provide a useful signal to estimate partisanship (Hemphill et al., 2016).

Closely related to the concept of controversy and the “filter bubble effect” is the concept of bias (Baeza-Yates, 2018), which refers to the presentation of information according to the viewpoints or interests of the journalists and the news agencies. Detecting bias is very important to help users to acquire balanced information. Moreover, how a
piece of information is reported has the capacity to evoke different sentiments in the audience, which may have large social implications (especially in very controversial topics such as terror attacks and religion issues).

In this paper, we approach this very broad topic by focusing on the problem of detecting hyperpartisan news, namely news written with an extreme manipulation of the reality on the basis of an underlying, typically extreme, ideology. This problem has received little attention in the context of the automatic detection of fake news, despite the potential correlation between them. Seminal work from (Potthast et al., 2018) presents a comparative style analysis of hyperpartisan news, evaluating features such as characters n-grams, stop words, part-of-speech, readability scores, and ratios of quoted words and external links. The results indicate that a topic-based model outperforms a style-based one to separate the left, right and mainstream orientations.

More recently, in (Kiesel et al., 2019), the features that participants used in SemEval-2019 task 4 on hyperpartisan news detection have been summarized: n-grams, word embeddings, stylometry (e.g., punctuation and article structure), sentiment and emotion features, named entities, quotations, hyperlinks, and publication date. Using the same dataset from SemEval-2019, (Anthonio, 2019) evaluated features like bag-of-words, bag-of-clusters, word embeddings and contextual character-based embeddings, POS n-grams, stylistic features and the sentiment; the authors found that dense document representations work better across domains and tasks than traditional sparse representations. Finally, (Hosseinia, 2020) found effective to use personality information in hyperpartisan news detection after topic-based sub-sampling of the news training data. The datasets proposed in (Kiesel et al., 2019) were manually labeled and the largest one was labeled in a semi-automated manner via distant supervision.

Instead of employing the datasets from (Kiesel et al., 2019), we build upon previous work and use the dataset from (Potthast et al., 2018): this way we can investigate hyperpartisan-biased news (i.e., extremely one-sided) that have been manually fact-checked by journalists from Buzzfeed, and contrast our results with what they achieved. The articles originated from 9 well-known political publishers, three each from the mainstream, the hyperpartisan left-wing, and the hyperpartisan right-wing. To detect hyperpartisanship, we aim to explore the trade-off between the performance of the models and the transparency of their results. Taking this into account, we apply two approaches diametrically opposite to each other in the text classification state of the art. On the one hand, we use three transformer-based models, which have shown outstanding performance, but high complexity and lack of transparency. On the other hand, we use a masking-based model that requires fewer computational-resources and showed a good performance in related tasks such as authorship attribution (Stamatatos, 2017a).

The masking technique transforms the original texts in a form where the textual structure is maintained, while letting the learning algorithm focus on the writing style or the topic-related information. This technique makes it possible for us to corroborate previous results that content matters more than style. Moreover, we aim to find explainable predictions of hyperpartisanship with the attention mechanism of the transformer-based models. With this purpose, we expect to derive the explanation by investigating the scores of different features used to output the final prediction. Based on this, we contrast the transparency of both approaches by comparing the relevant parts of the texts that they highlight.

The rest of the paper is structured as follows. In Section 2 we describe our method to hyperpartisan news detection based on masking. Section 3 presents details on the dataset and the experimental setup. In Section 4 we show the obtained results and discuss about them. Finally, Section 5 concludes with some directions for future work.

2 Masking and Transformer-based Models

2.1 Investigating Masking for Hyperpartisanship Detection

The masking technique that we propose here for the hyperpartisan news detection task has been applied to text clustering (Granados et al., 2011), authorship attribution (Stamatatos, 2017a), and deception detection (Sánchez-Junquera, 2018) with encouraging results. The main idea of the proposed method is to transform the original texts to a form where the textual structure, related to a general style (or topic), is maintained while content-related (or style-related) words are masked. To this end, all the occurrences of non-desired terms are re-
placed by symbols. Let \( W_k \) be the set of the \( k \) most frequent words, we mask all the occurrences of a word \( w \in W_k \) if we want to learn a topic-related model, or we mask all \( w \notin W_k \) if we want to learn a style-based model. Whatever the case, the way in which we mask the terms in this work is called Distorted View with Single Asterisks and consists in replacing \( w \) with a single asterisk or a single # symbol if the term is a word or a number, respectively. For further masking methods, refer to (Stamatatos, 2017a).

Table 1 shows a fragment of an original text and the result of masking style-related information or topic-related information. With the former we obtain distorted texts that allow for learning a topic-based model; on the other hand, with the latter, it is possible to learn a style-based model. One of the options to choose the terms to be masked or maintained without masking is to take the most frequent words of the target language (Stamatatos, 2017a). In the original text from the table, we highlight some of the most frequent words in English.

### 2.2 Transformer-based Models

Transformer-based models have been trained with huge general language datasets. Such is the case of the Bidirectional Encoder Representations from Transformers (BERT). BERT is designed to pretrain deep bidirectional representations from an unlabeled text by jointly conditioning on both left and right context in all layers (Devlin et al., 2018). This text representation allows the model to capture complex patterns going beyond merely the use of words and capturing semantic and syntactic patterns in the same representation.

The framework of BERT consists of two steps: pre-training and fine-tuning. For the pre-training, the collected data included BooksCorpus (800M words) and English Wikipedia (2,500M words). The BERT_{BASE} model has 12 layers with 12 self-attention heads, and uses 768 as hidden size, with a total of 110M parameters; and the BERT_{LARGE} model has 24 layers with 16 self-attention heads, and uses 1024 as hidden size, with a total of 340M parameters. The vocabulary contains 30K tokens. For fine-tuning, the model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream task, which in our case are 1555 news annotated with the political orientation. The first token of every sequence is always a special classification token ([CLS]), which is used as the aggregate sequence representation for classification tasks. In our work, we add to the [CLS] representation two dense layers and a Softmax function to obtain the binary classification.

In this paper we evaluate three transformer-based models: BERT; the multilingual BERT (M-BERT) (Devlin et al., 2018), which was pretrained on the concatenation of monolingual Wikipedia datasets from 104 languages (Wang et al., 2019; Pires et al., 2019); and XLM-RoBERTa, which was pretrained on 2.5TB of newly created clean CommonCrawl data in 100 languages (Cañete et al., 2020).

### 3 Experiments

We used the BuzzedFeed-Webis Fake News Corpus 2016 collected by (Potthast et al., 2018) whose articles were labeled with respect to three political orientations: mainstream, left-wing, and right-wing (see Table 2). Each article was taken from one of 9 publishers known as hyperpartisan left/right or mainstream in a period close to the US presidential elections of 2016. Therefore, the content of all the articles is related to the same topic. During initial data analysis and prototyping we identified a variety of issues with the original dataset: we cleaned the data excluding articles with empty or bogus texts, duplicates. As a result, we obtained a new dataset with 1555 articles out of 1627.1 Following the settings of (Potthast et al., 2018), we balanced the training set using random duplicate oversampling.

#### 3.1 Masking Content vs. Style in Hyperpartisan News

In this section, we reported the results of the masking technique from two different perspectives. In one setting, we masked topic-related information in order to maintain the predominant writing style used in each orientation. We call this approach a style-based model. With that intention we selected the \( k \) most frequent words from the target language, and then we transformed the texts by masking the occurrences of the rest of the words. In another setting, we masked style-related information to allow the system to focus only on the topic-related differences between the orientations. We call this a topic-based model. For this, we masked the \( k \) most frequent words and maintained intact the rest.

\[^1\text{The dataset will be available to the research community upon acceptance.}\]
Table 1: Examples of masking style-related information or topic-related information.

<table>
<thead>
<tr>
<th>Original text</th>
<th>Masking topic-related words</th>
<th>Masking style-related words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Officers went after Christopher Few after watching an argument between him and his girlfriend outside a bar just before the 2015 shooting</td>
<td>* went after * Few after * an * between him and his * a * just before the # *</td>
<td>Officers * * Christopher * * watching * argument * * * girlfriend outside * bar * * * 2015 shooting</td>
</tr>
</tbody>
</table>

Table 2: Statistics of the original dataset and its subset used in this paper.

<table>
<thead>
<tr>
<th></th>
<th>Left-wing</th>
<th>Mainstream</th>
<th>Right-wing</th>
<th>( \Sigma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original data (Potthast et al., 2018)</td>
<td>256</td>
<td>826</td>
<td>545</td>
<td>1627</td>
</tr>
<tr>
<td>Cleaned data</td>
<td>252</td>
<td>787</td>
<td>516</td>
<td>1555</td>
</tr>
</tbody>
</table>

After the text transformation by the masking process in both the training and test sets, we represented the documents with character n-grams and compared the results obtained with the style-based and the topic-related models.

3.2 Experimental Setup

Text Transformation: We evaluated different values of \( k \) \((k \in \{100, 200, ..., 5000\})\) for extracting the \( k \) most frequent words from English\(^2\). For the comparison of the results obtained by each model with the ones of the state of the art, we only showed the results fixing \( k = 500 \).

Text Representation: We used a standard bag-of-words representation with \( tf \) weighting and extracted character 5-grams with a frequency lower than 50.

Classifiers: We compared the results obtained with Naïve Bayes (NB), Support Vector Machine (SVM) and Random Forest (RF); for the three classifiers we used the versions implemented in `sklearn` with the parameters set by default.

Transformers: We approached the hyperparameter tuning by grid search. The best results were obtained with: \( learning\ rate = 3e^{-5}; \) the \( batch\ size = 16; \) and the \( adam\) optimizer. Moreover, we applied a dropout value of 0.3 to the last dense layer. We have selected a value of 200 for the \( max\_length\) hyperparameter.

Evaluation: We performed 3-fold cross-validation with the same configuration used in (Potthast et al., 2018). Therefore, each fold comprised one publisher from each orientation (the classifiers did not learn a publisher’s style). We used macro \( F_1 \) as the evaluation measure since the test set is unbalanced with respect to the three classes. In order to compare our results with those reported in (Potthast et al., 2018), we also used accuracy, precision, and recall.

Baseline: Our baseline method is based on the same text representation with the character n-grams features, but without masking any word.

4 Results and Discussion

Table 3 shows the results of the proposed method and the system from (Potthast et al., 2018)\(^3\) in our cleaned dataset (Section 3), both considering topic and style-based methods. In order to compare our results with those reported in (Potthast et al., 2018), we report the same measures the authors used. We also include the macro \( F_1 \) score because of the unbalance test set. For these experiments we extract the character 5-grams from the transformed texts, taking into account that as more narrow is the domain more sense has the use of longer n-grams. We follow the steps of (Stamatatos, 2017b) and set \( k = 500 \) for this comparison results.

Similar to (Potthast et al., 2018), the topic-based model achieves better results than the style-related model. However, the differences between the results of the two evaluated approaches are much higher (0.66 vs. 0.57 according to Macro \( F_1 \)) than those obtained from the system of (Potthast et al., 2018) (0.63 vs. 0.61). The highest scores of the masking technique were consistently achieved

\(^2\)We use the BNC corpus (https://www.kilgarriff.co.uk/bnc-readme.html) for the extraction of the most frequent words as in (Stamatatos, 2017a).

\(^3\)https://github.com/webis-de/ACL-18
using the SVM classifier and masking the style-related information (i.e., applying the topic-related model). This could be explained with the fact that all the articles are about the same political event in a very limited period of time. In line with what was already pointed out in Potthast et al. (2018), the left-wing orientation is harder to predict, possibly because this class is represented with fewer examples in the dataset.

Another reason why our masking approach achieves better results than the system from Potthast et al. (2018), could be that we use a higher length of character n-grams. In fact, comparing their results against our baseline model, it is possible to note that even without masking any word, the classifier obtains better results. This suggests that the good results are due to the length of the character n-grams rather than the use of the masking technique.

The last three rows of Table 3 show the results of the transformer-based models. As we can see, these models achieved the highest results, in particular the BERT model, with a Macro $F_1$ = 0.86. These models are known for their ability to capture complex syntactic and semantic patterns, therefore, these results are somehow justified to be the highest compared to the masking approach. However, what is interesting at this point is the effectiveness of the models at predicting the correct orientation using just the beginning of the news (max Length = 200). This is aligned to the work of Ghanem et al. (2021) that focused on analyzing the initial part of false news articles. The authors as-

Table 3: Results of the proposed masking technique ($k = 500$ and $n = 5$) applied to mask topic-related information or style-related information. NB: Naive Bayes; RF: Random Forest; SVM: Support Vector Machine. The last two rows show the results obtained by applying the system from Potthast et al. (2018) to our cleaned dataset (Section 3).

<table>
<thead>
<tr>
<th>Masking Method</th>
<th>Classifier</th>
<th>Macro F1</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline model</td>
<td>NB</td>
<td>0.52</td>
<td>0.56</td>
<td>0.28</td>
<td>0.57</td>
<td>0.81</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0.56</td>
<td>0.62</td>
<td>0.28</td>
<td>0.61</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>0.70</td>
<td>0.77</td>
<td>0.55</td>
<td>0.75</td>
<td>0.84</td>
</tr>
<tr>
<td>Style-based model</td>
<td>RF</td>
<td>0.46</td>
<td>0.53</td>
<td>0.24</td>
<td>0.58</td>
<td>0.64</td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>0.57</td>
<td>0.66</td>
<td>0.33</td>
<td>0.66</td>
<td>0.75</td>
</tr>
<tr>
<td>Topic-based model</td>
<td>NB</td>
<td>0.54</td>
<td>0.60</td>
<td>0.27</td>
<td>0.63</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0.53</td>
<td>0.55</td>
<td>0.27</td>
<td>0.64</td>
<td>0.71</td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>0.66</td>
<td>0.74</td>
<td>0.48</td>
<td>0.73</td>
<td>0.81</td>
</tr>
</tbody>
</table>

System from Potthast et al. (2018) (applied to our cleaned dataset)

<table>
<thead>
<tr>
<th>Transformer-based models</th>
<th>Macro F1</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-BERT</td>
<td>0.76</td>
<td>0.83</td>
<td>0.65</td>
<td>0.75</td>
<td>0.93</td>
</tr>
<tr>
<td>XLM-RoBERTa</td>
<td>0.80</td>
<td>0.86</td>
<td>0.80</td>
<td>0.76</td>
<td>0.95</td>
</tr>
<tr>
<td>BERT</td>
<td>0.86</td>
<td>0.89</td>
<td>0.77</td>
<td>0.87</td>
<td>0.94</td>
</tr>
</tbody>
</table>

Table 4: Most relevant features to each class.

Table 4 shows the features with the highest weights from the SVM (we used scikit-learn’s method to collect feature weights). It is possible to note that the mention of cnn was learned as a discriminative feature when the news from that publisher were used in the training (in the topic-based model). However, this feature is infrequent in the test set where no news from CNN publisher was included.

The features related to Donald Trump (donal and onal), and Hillary Clinton (illary and illary) are more frequent in one of the hyperpartisan orientation, and none of them occurs frequently in the
main
Obama proposed a plan which his son pretty much confirmed in a foolish statement. The content of those tax returns has been the subject of much speculation, but given Trump’s long history of tax evasion and political bribery, it doesn’t take much imagination to assume he’s committing some kind of fraud.

right
The media, which had started by March 16, 2011, and shared with POLITICO, reads: Jim, on Kenya your person in the field might look into the impact there of Obama’s public comments about his father. I’m told by State Dept officials that Obama publicly derided his father on ( ...). Blumenthal, a longtime confidant of both Bill and Hillary Clinton, emerged as a frequent correspondent in the former secretary of ( ...).

Table 5: Fragments of original texts and their transformation by masking the k most frequent terms. Some of the features from Table 4 using the topic-related model are highlighted.

4.2 Features with the Highest Attention Scores
Transformer-based models allow us to visualize different parts of the news according to the scores they received to obtain the final prediction. In Figure 1 we show examples of news predicted correctly by BERT (the model with the highest $F_1$ score). Due to space limitations, we provide fragments of six news, two per orientation. The more intense the color, the greater is the weight of attention given by the model.

In the examples from 1a, the left-wing orientation remarks the names of the opposite politicians, and it is possible to see which of them is the favourite of the journalist. In particular, the leader of the right-wing (i.e., Trump) is referred in a negative way (he does not know his own words) while Hillary Clinton, the representative of the left-wing, is favored by the news. Similar to this, examples 1c do the same but in the opposite direction; i.e., Hillary Clinton is put as a very negative “character” who loves taxes and is the most despicable liar ever. However, examples from 1c offer a comparison in which keep the reader in a neutral position. Moreover, in the second mainstream news, Trump’s campaign is mentioned without describing the stance of the author whether Trump did well or not in his topic selection. This suggests that the style used to speak about the leaders can differ from the more biased (hyperpartisan) news to the less biased (mainstream).

We can conclude that the attention mechanism of the transformers not only help in doing effective predictions, but offer some extra information that could be useful to understand some insights about hyperpartisanship. For example, the words with the highest scores can be used in other strategies to confirm the previous results that topic-based models outperform a style-based one at distinguishing left, right and mainstream orientations (Potthast et al., 2018).

5 Conclusions
In this paper we presented initial experiments on the task of hyperpartisan news detection. In particular, we aimed to explore the trade-off between performance and transparency, and proposed a comparison of two different approaches. First, we explored the use of masking techniques to boost the performance of a lexicalized classifier. Our results corroborate previous research on the importance of content features to detect extreme content: masking, in addition, shows the benefits of reducing data sparsity for this task comparing our results with the state of the art. We evaluated different values of the parameters and see that finally our baseline model, in which we extract character 5-grams without applying any masking process, achieves the better results. This seems to indicate a strong lexical overlap between different sources with the same orientation, which, in turn, calls for more challenging datasets and task formulations to encourage the development of models covering more subtle, i.e., implicit, forms of bias. Future datasets could consider more topics and different time spans to avoid the models learn from the topic, rather than the target classes.

Second, we used three transformer-based models (BERT, M-BERT, and XLM-RoBERTa) that are resource-hungrier than the masking technique, and achieved the highest results. We also presented some examples of how these models, through their attention scores, provide additional information about the relevant parts of the text for distinguishing their political orientation. Considering the high effectiveness of these models, and that they only observe the first part of the news, we will evaluate as future work how necessary is to use all the
Hyperpartisan (left-wing) news.

Hyperpartisan (right-wing) news.

Non-hyperpartisan (mainstream orientation) news.

News (and not only the beginning), e.g., with the Transformer-XL model (Dai et al., 2019). Moreover, we are motivated to take advantage of the attention scores to study in more detail the style used in hyperpartisan news in order to improve the predictions.

Acknowledgments

The work of the authors from the Universitat Politècnica de València was funded by the Spanish Ministry of Science and Innovation under the research project MISMIS-FAKEnHATE on MIS-information and MIScommunication in social media: FAKE news and HATE speech (PGC2018-096212-B-C31). Experiments were carried out on the GPU cluster at PRHLT thanks to the PROMETEO/2019/121 (DeepPattern) research project funded by the Generalitat Valenciana.

References


Master’s thesis, University of the Basque Country UPV/EHU.


Ana Granados, Manuel Cebrian, David Camacho, and Francisco de Borja Rodriguez. 2011. Reducing the loss of information through annealing text distortion. IEEE Transactions on Knowledge and Data Engineering, 23(7):1090–1102.


Serbian NER&Beyond: The Archaic and the Modern Intertwinned

Branislava Šandrih Todorović
University of Belgrade
Faculty of Philology
branislava.sandrih@fil.bg.ac.rs

Ranka Stanković
University of Belgrade
Faculty of Mining and Geology
ranka.stankovic@rgf.bg.ac.rs

Cvetana Krstev
University of Belgrade
Faculty of Philology
cvetana@matf.bg.ac.rs

Milica Ikonić Nešić
University of Belgrade
Faculty of Philology
milica.ikonic.nesic@fil.bg.ac.rs

Abstract

In this work, we present a Serbian literary corpus that is being developed under the umbrella of the “Distant Reading for European Literary History” COST Action CA16204. Using this corpus of novels written more than a century ago, we have developed and made publicly available a Named Entity Recognizer (NER) trained to recognize 7 different named entity types, with a Convolutional Neural Network (CNN) architecture, having $F_1$ score of $≈91\%$ on the test dataset. This model has been further assessed on a separate evaluation dataset. We wrap up with comparison of the developed model with the existing one, followed by a discussion of pros and cons of the both models.

1 Introduction

The “Distant Reading for European Literary History”\(^1\) (COST Action CA16204) has started in 2017 with the purpose of using computational methods to analyse large collections of literary texts (Stanković et al., 2019; Frontini et al., 2020). The main goal of this ongoing action is to compile a multilingual open-source collection, named European Literary Text Collection (ELTeC), containing linguistically annotated sub-collections of 100 novels per language written more than 100 years ago.

In this paper, we present a collection of Serbian texts in this corpus, named SrPTELTeC. Alongside, we describe our efforts in developing its Named Entity (NE) layer, defined previously as one of the main action’s deliverables.

For this purpose, we adjusted and used the existing rule-based NE recognizer for Serbian, dubbed SrPNER, that we will describe in Section 2 together with some approaches to NE recognition in literary texts. This SrPNER model was applied to the raw version of the selected texts from SrPTELTeC collection, presented in Section 3. Based on the specifically tailored guidelines, different evaluators performed careful checks and corrections, yielding a gold standard (SrPTELTeC-GOLD). This enabled us to train a CNN-based NE recognizer, named SrPCNNER, presented in Section 4. Having the gold dataset, prepared as described in Subsection 4.1, we trained (Subsection 4.2) and evaluated the model in two different settings: first, we discussed our model’s performance on the SrPTELTeC-GOLD test subset, as shown in Subsection 4.3. Afterwards we carried out a detailed evaluation on a collection of novels that were not present in the gold standard, named SrPTELTeC-EVAL, with the findings and a thorough discussion given in Section 5. Finally, conclusions and plans for the future work were stated in Section 6.

2 Related Work

The existence of large-scale lexical resources for Serbian, e-dictionaries in particular (Krstev, 2008), coupled with local grammars in the form of finite-state transducers (Vitas and Krstev, 2012), enabled the development of a comprehensive rule-based system for NER SrPNER. This system presented by Krstev et al. (2014) targeted 11 classes of NEs: dates and time (moments and periods), money and measurement expressions, geopolitical names (countries, settlements, oronyms and hydronyms), and personal names (one or more last names with or without first names and nicknames). The system was developed to recognize NEs in

---

\(^1\)Distant Reading,
https://www.distant-reading.net

Proceedings of Recent Advances in Natural Language Processing, pages 1256–1264
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_142
newspapers and similar texts. It was manually evaluated on a sample of unseen newspaper texts. The overall $F_1$ score of the model was $\approx 96\%$. To the best of our knowledge, so far there were no attempts to produce a NER system for Serbian literary texts.

The enhanced version of SrPNER was later utilized by Šandrih et al. (2019) for the preparation of a gold standard annotated with personal names, which was used for building training sets for 4 different levels of annotation, on which two ML-based NE recognizers were trained and evaluated (SpaCy and Stanford). As a support for the developed NER models, Šandrih et al. (2019) joined several existing tools and developed various new tools, combined into a web platform NER&Beyond.\(^2\)

Although NER systems in general were developed mostly for newspaper and similar texts, there were some endeavours to produce functional systems for literary texts as well. Enrichment of French Renaissance texts with proper names (Maurel et al., 2014) faced two challenges: text diversity due to various spellings of words, and need to deal with numerous XML-TEI tags used to preserve the format of original editions. Authors’ solution was based on the cascades of finite-state automata and both general dictionaries and those built specifically for the project. The evaluation showed that the slot error rate of name tagging was 6.1%.

A dataset of literary entities comprising 210,532 tokens evenly drawn from 100 different English literary texts annotated with ACE entity categories (person, location, geo-political entity, facility, organization, and vehicle)\(^3\) was published in (Bamman et al., 2019). The authors’ main motivation was to asses NER models’ performance on different types of texts. Their conclusion was that recognition improved for almost all entity types when literary texts were used for the both training and evaluation (on average $P = 75.1\%$, $R = 62.6\%$ and $F_1 = 68.3\%$), whilst for training on general texts, such as news data, and testing on literary texts the results were much poorer (on average $P = 57.8\%$, $R = 37.7\%$ and $F_1 = 45.7\%$).

SHINRA2020-ML shared-task (Sekine et al., 2020) targeted the categorization of Wikipedia entities using the Extended Named Entity (ENE) hierarchy in 30 languages (Serbian was not one of them). ENE included about 220 fine-grained categories of NEs in a hierarchy of up to four layers. Some traditional NE types such as location were specified as either geopolitical location (“city”, “province”, “country”, etc.) or geological region (“mountain”, “river”, “lake”, etc.). ENE also included some new NE types like “products”, “event”, “position”, etc.

Dekker et al. (2019) experimented with different off-the-shelf NER tools for the extraction of social network graphs from classic and modern English fiction novels. The authors wanted to find out to what extent are these tools suitable for identifying fictional characters in novels, and what are differences and similarities that can be discovered between social networks extracted for different novels.

Distant Reading Training School for Named Entity Recognition and Geo-Tagging for Literary Analysis organized within the COST Action 16204\(^4\) covered NER approaches in general, annotation campaigns, practical work with NER tools, annotating NER in TEI, analyzing NER annotation for literary characters and place names and NER data analysis. Different types of NER systems were tested for several languages, some based on symbolic methods, relying on rules developed by experts and dictionaries (gazetteers), others using statistical and data-driven approach.

The NE layer of ELTeC corpus has presently been produced for three languages: Hungarian, Portuguese, Slovene. Santos et al. (2020) reported on the NER annotation of the Portuguese sub-collection of the ELTeC corpus. Authors used the PALAVRAS-NER parser, a Constraint Grammar (CG) system, in which NER is an integrated task of grammatical tagging, implemented with the basic tagset of 6 NE categories (person, organization, place, event, semantic products and objects) with about 20 subcategories at three levels, disambiguated by CG-rules: known lexical entries and gazetteer lists, pattern-based name type prediction and context-based name type inference for unkno-

\(^2\)NER&Beyond, \url{http://nerbeyond.jerteh.rs/}

\(^3\)ACE (Automatic Content Extraction) 2005 Multilingual Training Corpus, \url{https://catalog.ldc.upenn.edu/LDC2006T06}

\(^4\)Materials for the NER Training School, \url{https://github.com/distantreading/WG2/tree/master/NER_TS}
This system was applied to eight novels that were fully human revised. Evaluation results varied for precision from 64.6% to 80.8%, and recall from 64.3% to 82.0%.

At the mentioned Distant reading training school it was concluded that spaCy module\(^5\) for Python was used for training NER models for many involved languages, already having tagsets that could be mapped to the ELTeC annotation scheme, elaborated later in Section 3. Partalidou et al. (2019) developed a POS-tagger and a NER for Greek using spaCy, based on newspaper articles and Wikipedia dataset, able to recognize the following entity types: location, organization, person and facility. Jabbari et al. (2020) created a corpus consisting of news articles in French, which served as a dataset for training and evaluation of a NER and a relation extraction algorithms using spaCy. Modrzejewski et al. (2020) incorporated NER trained in spaCy into an English/German Machine Translation system, with the aim to improve NE translation.

Moreover, Jiang et al. (2016) conducted a comparative evaluation of different publicly available NER tools. Based on different criteria, authors concluded that spaCy was among best performing across all tested datasets. Having all this in mind, we decided for spaCy as a framework for developing a Serbian NER model on a collection old literary texts.

### 3 Serbian Collection in the ELTeC

As described earlier in Section 1, the focus of the COST Action CA16204 is to compile the ELTeC corpus containing collections of old European novels published between 1840 and 1920 in various languages. In order to make these sub-collections decent representatives of their corresponding languages, the novels were selected to evenly represent a) novels of various sizes: short, medium, long; b) four twenty-year time periods within the examined time span, c) canonical novels as well as those not known to wider audience or completely forgotten, as judged by the number of reprints, and d) female and male authors (Frontini et al., 2020).

The last version of the ELTeC (v. 1.1.0) was released in April 2021.\(^6\) It contained 14 language sub-collections each with at least 50 novels, while 8 collections contained targeted 100 novels per language.

The SrpELTeC corpus\(^7\) in the latest ELTeC release has 90 novels. The work on this collection is still in progress with the aim to obtain the complete collection by the end of the project. Contrary to a number of other European languages involved in this action, the Serbian corpus is being produced from scratch, because the vast majority of novels from the selected time period were not digitized before, they were not digitized in the proper manner or were not available (Krstev et al., 2019).

This preparation procedure involved several steps: selection of novels, retrieval of hard copies, scanning, OCR, automatic correction of OCR errors (for which a specialized tool based on the Serbian morphological dictionaries was produced (Krstev and Stanković, 2020)), correction of remaining errors by a number of volunteer readers, and production of metadata.

One of the important aspects of this ELTeC collection is to feature annotations of certain named entities. At this moment, annotation of named entities is carried out for nine languages, including Serbian. According to the guidelines, the common NER tagset includes the following 7 categories: demonyms (DEMO), professions and titles (ROLE), works of art (WORK), person names (PERS), places (LOC), events (EVENT) and organizations (ORG).\(^8\)

### 4 SrpCNNER Model for Serbian

In this section we first explain how we have turned the SrpELTeC corpus into a dataset for NER. Afterwards, we describe the training of the NER model SrpCNNER, followed by a detailed evaluation. Web users can navigate to http://ner.jerteh.rs/ in order to apply the SrpCNNER model directly on input text. The model can also be applied to a custom-size collection of text files using the previously mentioned NER&Beyond web platform.

---

5spaCy, https://spacy.io/

6ELTeC (Distant Reading for European Literary Hi-

7SrELTeC, https://distantreading.github.io/ELTeC/srp/index.html

8ELTeC Collections with NE-annotations, http://brat.jerteh.rs/index.xhtml#/eltec/
4.1 Gold Standard: SrpELTeC-gold

The SrpNER system for Serbian introduced in Section 2 was used in the first stage of the gold standard preparation (dubbed SrpELTeC-gold) in order to automatically annotate Sr- pELTeC collection. The tagset used by SrpNER differed from the simplified tagset used in the ELTeC project – the tags are more refined, e.g. toponyms are classified as oronyms, hydronyms, settlements etc., and nesting of tags is allowed. Thus, the tags produced by SrpNER had to be mapped to ElTeC tags as illustrated in Figure 1:

![Figure 1](image)

Before text annotation, we used the advantage of rule-based NER systems and adjusted SrpNER to these specific texts that differ significantly from newspaper texts for which SrpNER was primarily developed in order to improve its performance and facilitate the work of evaluators. Some modifications of rules and used lexicons were done for the whole collection (e.g. Danas ‘today’ cannot be the name of an organization since this publishing house was established 20 years ago), while others were novel-specific (e.g. Una can be the first name or the name of the river – we retained only the possibility appropriate to the particular novel).

The EVENT named entity is somewhat special: SrpNER does not recognize this entity, so the evaluators were asked to identify and annotate them when they occur in text. SrpNER does not recognize WORK entity either, but these annotations were in many cases added by volunteer readers during text correction.

Afterwards, students were given different novel chapters along with the annotation guidelines presented briefly in Table 1. Following these instructions and under constant supervision of their professors, students manually corrected the automatically annotated chapters.

The evaluators were divided into two groups: the first group performed corrections using the BRAT annotation tool,\(^9\) while the second group used the INCEpTION.\(^10\) We wanted to receive user feedback on both platforms for the sake of creating the annotation process as comfortable and efficient as possible in the future, but also to provide choice to annotators. The fundamental difference was the input format these platforms needed: BRAT tool uses the standoff format, whilst INCEpTION relies on the CoNLL-2002 verticalized format.\(^11\) In order to convert from one format into another, we used the NERBeyond web application.

Table 2 displays distribution of different entity types over SrpELTeC-gold novels. The first four digits of text identifiers represent the year of the first publication of a novel. For some novels, NER was not performed on the whole text, but rather on randomly selected chapters. These annotated samples were also included in the gold standard. The cumulative values of entities on all samples are indicated in the first row (ID “sample”). Column $\sum_{tok}$ indicates a novel’s size in terms of tokens.

4.2 Training

We trained our SrPCNNER model on the SrpELTEC-gold corpus using the spaCy Python module, version 3.0. In order to prepare the dataset for training, we first segmented texts into sentences, ending up with 43,129 sentences in total, including sentences that did not contain named entities. Afterwards, we randomly shuffled and split these sentences into training, test and development sets with the ratio of 8:1:1, i.e. 34,503 sentences in the train set, and the same number of sentences, 4,313, in the test and development sets, respectively. These sentences were prepared as Python list-objects containing tuples as elements. An example of such tuple is the following:

```
"Hadži-era je za vreme ušao u sobu agama, da im nazove dobro jutro, a manastir-
```

\(^9\)BRAT, [https://brat.nlplab.org](https://brat.nlplab.org)

\(^10\)INCEpTION annotation tool, [https://inception-project.github.io/](https://inception-project.github.io/)

\(^11\)Among other CoNLL and XML variants that this tool supports.
Entity | Explanation
--- | ---
PERS | Personal names First names, surnames, nicknames and their combinations (of real people and fictional characters, including gods and saints). Possessive adjectives from personal names should not be annotated.
ROLE | Occupations and titles Occupations, titles and responsibilities: doctor, teacher; king; director.
LOC | Locations Continents, countries, regions, populated places, onyms, water surfaces, names of celestial bodies, city locations.
DEMO | Origin or residence Residents of states, cities, regions, or ethnic groups; adjectives derived from the names of locations.
ORG | Organizations, institutions, societies Company names, politic parties, educational institutions, sport teams, hospitals, museums, libraries, hotels, cafes, churches and shrines.
WORK | Art works Titles of books, plays, poems, paintings, sculptures, newspapers.
EVENT | Events Names of events that are repeated regularly or have happened once but have their own name: natural disasters, revolutions, battles, wars.

Table 1: Annotation guidelines.

<table>
<thead>
<tr>
<th>ID</th>
<th>PERS</th>
<th>ROLE</th>
<th>LOC</th>
<th>DEMO</th>
<th>ORG</th>
<th>WORK</th>
<th>EVENT</th>
<th>Σ_tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>samples</td>
<td>707</td>
<td>207</td>
<td>156</td>
<td>105</td>
<td>8</td>
<td>4</td>
<td>14</td>
<td>19,274</td>
</tr>
<tr>
<td>18750</td>
<td>1,688</td>
<td>1,050</td>
<td>388</td>
<td>239</td>
<td>29</td>
<td>10</td>
<td>21</td>
<td>31,743</td>
</tr>
<tr>
<td>18871</td>
<td>1,612</td>
<td>1,509</td>
<td>328</td>
<td>229</td>
<td>52</td>
<td>60</td>
<td>18</td>
<td>34,324</td>
</tr>
<tr>
<td>18880</td>
<td>1,372</td>
<td>986</td>
<td>271</td>
<td>201</td>
<td>32</td>
<td>59</td>
<td>10</td>
<td>26,642</td>
</tr>
<tr>
<td>18881</td>
<td>935</td>
<td>619</td>
<td>95</td>
<td>105</td>
<td>12</td>
<td>14</td>
<td>1</td>
<td>13,898</td>
</tr>
<tr>
<td>18890</td>
<td>804</td>
<td>714</td>
<td>36</td>
<td>56</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>29,337</td>
</tr>
<tr>
<td>18932</td>
<td>1,521</td>
<td>259</td>
<td>46</td>
<td>35</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>16,821</td>
</tr>
<tr>
<td>18950</td>
<td>764</td>
<td>581</td>
<td>51</td>
<td>103</td>
<td>12</td>
<td>6</td>
<td>33</td>
<td>14,454</td>
</tr>
<tr>
<td>19021</td>
<td>1,647</td>
<td>2,285</td>
<td>123</td>
<td>58</td>
<td>82</td>
<td>4</td>
<td>15</td>
<td>40,804</td>
</tr>
<tr>
<td>19040</td>
<td>1,655</td>
<td>917</td>
<td>221</td>
<td>281</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>32,367</td>
</tr>
<tr>
<td>19140</td>
<td>770</td>
<td>412</td>
<td>240</td>
<td>94</td>
<td>45</td>
<td>5</td>
<td>7</td>
<td>31,583</td>
</tr>
<tr>
<td>19190</td>
<td>1,181</td>
<td>797</td>
<td>8</td>
<td>13</td>
<td>49</td>
<td>24</td>
<td>19</td>
<td>33,562</td>
</tr>
<tr>
<td>total</td>
<td>14,788</td>
<td>10,405</td>
<td>1,979</td>
<td>1,568</td>
<td>323</td>
<td>198</td>
<td>149</td>
<td>330,119</td>
</tr>
</tbody>
</table>

Table 2: SrPELT3C-Gold NE distribution.

ski [sluga] poče prishuživati rakiju i kaflu."\(^{12}\), entities: [(0, 10, ‘PERS’), (39, 44, ‘ROLE’), (86, 91, ‘ROLE’)]

The spaCy v3.0 enables specification of custom neural network architecture within a simple text file. Using the quick-start widget,\(^{13}\) user can easily set up the default setting configuration. In our case, the model’s language was

---

\(^{12}\)Translates as: *In the meantime, Haji- era entered the room to wish agas good morning, when the monastery servant started offering coffee and brandy.*

\(^{13}\)Quick-start spaCy3 widget, [https://spacy.io/usage/training#quickstart](https://spacy.io/usage/training#quickstart)
the architecture (architecture) to HashEmbedCNN\(^\text{14}\) having width of the input and the output equal to 300 (width), with 8 convolutional layers (depth), 10,000 rows in the hash embedding tables (embed_size), with the recommended 1 token on either side to concatenate during the convolutions (window_size), without pretrained static vectors (pretrained_vectors = null).

Model training ended up after 11 epochs (the number of epochs is automatically generated), having 93.33\%, 90.14\% and 91.71\% F\(_1\) score, precision and recall, on the development set, respectively.

### 4.3 Evaluation

Afterwards, we examined our model’s performance on the test set. We run the previously trained model on raw, non-annotated sentences from the SrpELTeC. After comparing the obtained annotations with the ones given in the test subset of the SrpELTeC-GOLD, we obtained the precision (P), recall (R) and F\(_1\) scores displayed in Table 3.

<table>
<thead>
<tr>
<th>Type</th>
<th>P</th>
<th>R</th>
<th>F(_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERS</td>
<td>0.953</td>
<td>0.936</td>
<td>0.944</td>
</tr>
<tr>
<td>ROLE</td>
<td>0.940</td>
<td>0.917</td>
<td>0.928</td>
</tr>
<tr>
<td>LOC</td>
<td>0.849</td>
<td>0.778</td>
<td>0.812</td>
</tr>
<tr>
<td>DEMO</td>
<td>0.781</td>
<td>0.758</td>
<td>0.769</td>
</tr>
<tr>
<td>ORG</td>
<td>0.903</td>
<td>0.368</td>
<td>0.523</td>
</tr>
<tr>
<td>WORK</td>
<td>0.324</td>
<td>0.343</td>
<td>0.333</td>
</tr>
<tr>
<td>EVENT</td>
<td>0.792</td>
<td>0.655</td>
<td>0.717</td>
</tr>
</tbody>
</table>

Table 3: SrpCNNER on the test set.

The normalized confusion matrix is given in Figure 2 (‘O’ represents tokens that are not NE). One can observe that WORK and EVENT were frequently missed or confused with PERS.

### 5 Separate Evaluation Set

Despite the encouraging results obtained on the SrpELTeC-GOLD, shown in Subsection 4.3, we wanted to further assess our model’s performance. For this purpose, we prepared an independent evaluation set, dubbed SrpELTeC-EVAL, containing corrected annotated chapters from three novels that were not included in the training procedure. Table 4 displays entity distribution over SrpELTeC-EVAL. Named entities are represented by their first letter (e.g. P represents PERS). It should be noted that the EVENT type did not occur in this dataset.

<table>
<thead>
<tr>
<th>ID</th>
<th>P</th>
<th>R</th>
<th>L</th>
<th>D</th>
<th>O</th>
<th>W</th>
<th>(\sum_{\text{tok}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>19070</td>
<td>44</td>
<td>55</td>
<td>23</td>
<td>23</td>
<td>3</td>
<td>0</td>
<td>2,027</td>
</tr>
<tr>
<td>19180</td>
<td>18</td>
<td>13</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>3,928</td>
</tr>
<tr>
<td>19121</td>
<td>33</td>
<td>18</td>
<td>14</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3,045</td>
</tr>
</tbody>
</table>

Table 4: SrpELTeC-EVAL NE distribution.

We applied the same evaluation procedure for the both recognizers. After running a them on SrpELTeC-EVAL, we took the strictest approach and differentiated between the following three situations:

- **[TP]** an entity is recognized exactly as it should, comparing to the gold standard (the text and the named entity types match – true positives);
- **[FP]** there are three cases here: 1) an entity is recognized, but not with the correct type (e.g. PERS mistaken for a ROLE); 2) an entity is recognized as a correct type but the scope is not correct (e.g only a first name is recognized as PERS, although a full name is given); or 3) model annotated something that is not present in the gold standard – false positives;
[FN] an entity present in the gold standard was not recognized – false negatives.

In the subsections that follow, we analyze the performances of our newly trained model SrpCNNER and the adjusted SrpNER on the SrpELTeC-EVAL corpus. Finally, we discuss their strengths and weaknesses and make certain statements about their applicability in different contexts and situations.

5.1 SrpCNNER vs. SrpELTeC-EVAL

The overall results for the SrpCNNER are displayed in the upper part of Table 5. As previously explained, for the case of FP, there is a specific situation that something was recognized, but not with the correct entity type. Such cases are indicated by the number in parentheses of the FP column (therefore, numbers TP, FN and the one given in parentheses from the FP column sum up to the total number of entities given in the \( \sum \) column in Table 4).

<table>
<thead>
<tr>
<th>ID</th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SrpCNNER vs. SrpELTeC-EVAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19070</td>
<td>25 (18)</td>
<td>80</td>
<td>0.538</td>
<td>0.385</td>
<td>0.448</td>
<td></td>
</tr>
<tr>
<td>19180</td>
<td>29 (4)</td>
<td>12</td>
<td>0.450</td>
<td>0.692</td>
<td>0.545</td>
<td></td>
</tr>
<tr>
<td>19121</td>
<td>23 (6)</td>
<td>27</td>
<td>0.540</td>
<td>0.557</td>
<td>0.548</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SrpNER vs. SrpELTeC-EVAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19070</td>
<td>5 (2)</td>
<td>18</td>
<td>0.948</td>
<td>0.877</td>
<td>0.911</td>
<td></td>
</tr>
<tr>
<td>19180</td>
<td>24 (2)</td>
<td>14</td>
<td>0.509</td>
<td>0.650</td>
<td>0.574</td>
<td></td>
</tr>
<tr>
<td>19121</td>
<td>15 (0)</td>
<td>20</td>
<td>0.758</td>
<td>0.701</td>
<td>0.729</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Evaluation results SrpELTeC-EVAL.

Values of precision (\( P \)), recall (\( R \)) and \( F_1 \) scores over each entity are shown in the upper part of Figure 3.

5.2 SrpNER vs. SrpELTeC-eval

The overall results for the SrpNER are displayed in the lower part of Table 5. Values of precision (\( P \)), recall (\( R \)) and \( F_1 \) scores over each entity are shown in the lower part of Figure 3.

From the obtained results it is obvious that SrpNER was not nearly as successful as when applied to newspaper texts. This could well be expected since each novel has its own specifics, and one cannot say that novels in general share some common language features, as newspapers do. Also, one can observe that results are very different for each of three samples; however, we cannot draw some firm conclusions, since the used samples were rather small.

5.3 Discussion

Based on the results shown in Figure 3 (upper part) and Table 5, it becomes obvious that SrpCNNER does not perform so well on unseen texts. In order to understand the reasons for that, we observed each and single case in isolation, which brought us to certain findings.

SrpCNNER performed rather well in recognizing personal names (e.g. Ana, Nikola, Gavrakovici, Ismail), roles and titles (e.g. car ‘tsar’, sultan, princeza ‘princess’, sveštenik ‘priest’), locations (e.g. Beograd, Paris, Niš), and demonyms (e.g. Švaba ‘German’ (pejorative), ruskom ‘Russian’, francuskom ‘French’). However, the number of FP cases was intriguing, due to the ambiguity of use. For example, the model recognized all occurrences of the word otac ‘father’ as a ROLE, although it can represent both a male parent (which according to the guidelines should not be annotated) and a priest (which should be annotated). Similar is the case with čika ‘uncle’, which in Serbian, when used before a personal name, has the meaning of mister/sir (familiarly). Both words are used rather frequently, and out of 33 false positives for the novel 19180, 13 were occurrences of exactly these two words.

The novel 19070 revealed some new weak points. For example, occurrences such as Fatiseultan, Ismail-beg and Ahmed-hafuz are specific to this novel and they represent a combination of a PERS-ROLE entities, a construction that is not usual in Serbian – ROLE PERS order is preferred. SrpCNNER recognized these two entities as a single PERS, WORK or LOC entity (among 43 false positives for the 19070, 7 were these names in various inflected forms), or did not recognize them at all (14 times).

We also noticed that some false positives were due to specific characteristics of texts. Namely, the orthography in the old novels was not stable, leading to incorrect occurrences (according to contemporary usage); for instance, the word gospode ‘god’ was considered, according to the decision of the evaluator, FP because written with the lower-case G, while the same word written with the upper-case G Gospode ‘God’ was found among the true positives.

It should be noted that in literary texts it is not always easy to decide what is the right type of an NE. For instance, in a sentence from...
19180: Sa Tolstojem sam se pomirila i obožavam ga za Anu Karenjinu ‘I reconciled with Tolstoy and I adore him for Anna Karenina’, *Ana Karenjina* can refer to the novel (WORK) or to its main character (PERS), and it is open to interpretation. Similarly, the names of saints (PERS) were sometimes difficult to distinguish from festivities that celebrate them (EVENT). One such example from 18950 is: *Mi slavimo Svetog Nikolu, ovog letnjeg.* ‘We celebrate Saint Nicolas, the one that comes in summer.’

Finally, we have noticed that our gold standard has flaws, introduced by evaluators, especially when facing some of the tricky cases mentioned before. It would have certainly been better if we could engage two evaluators for each text, but our human resources were limited.

Overall conclusion is that SrpCNNER performs satisfactorily on similar texts, which can be seen from the model’s performance on the test set displayed in Table 3. Since this collection of novels contains very diverse texts, both lexically and syntactically, SrpCNNER did not generalize that well on unseen texts.

6 Conclusions and Future Work

We presented the corpus of old Serbian novels, which served as a basis for training a CNN-based NER model SrpCNNER using the spaCy module’s framework for Python. After comparing this newly developed model for Serbian with the existing rule-based SrpNER, we came to the conclusion that the previously developed one performs better on this type of texts, due to its adaptability. However, it is not easy to set it up and use it, while the model trained in spaCy can be easily and efficiently applied to the large text collections, and there is still a lot of room for improvement. First of all we need to remove observed flaws from SrpELTeC-gold. Moreover, in the future we intend to use the pre-trained word embedding vectors instead of the default *tok2vec* layer.

The integration of POS-tagging and lemmatization with NER into TEI ELTeC level 2 schema\(^{15}\) is an ongoing activity, where a pipeline starts with SrpNER annotation, followed by POS-tagging and lemmatization by a TreeTagger (Schmid, 1999; Stanković et al., 2020). As a result, first 16 novels from SrpELTeC collection were annotated with POS, lemmas, and NE in a format agreed by the COST action.

Acknowledgments

This research was done in the scope of the COST action CA16204 “Distant Reading for European Literary History”. We thank the students of the Department of Library and Information Sciences, Faculty of Philology, master students of Social Sciences and Computing at Multidisciplinary Graduate Studies and PhD students of Intelligent systems program (University of Belgrade) for their help in evaluating the data.

\(^{15}\)Encoding Guidelines for the ELTeC: level 2, [https://distantreading.github.io/Schema/eltec--2.html](https://distantreading.github.io/Schema/eltec--2.html)
References


A Semi-Supervised Approach to Detect Toxic Comments

Ghivvago D. Saraiva1, Rafael T. Anchiêta2, Francisco A. R. Neto2 and Raimundo S. Moura1

1Federal University of Piauí, Brazil
{ghivvagodamas.ufpi,rsm}@ufpi.edu.br
2Federal Institute of Piauí, Brazil
{rta,farn}@ifpi.edu.br

Abstract

Toxic comments contain forms of non-acceptable language targeted towards groups or individuals. These types of comments become a serious concern for government organizations, online communities, and social media platforms. Although there are some approaches to handle non-acceptable language, most of them focus on supervised learning and the English language. In this paper, we deal with toxic comment detection as a semi-supervised strategy over a heterogeneous graph. We evaluate the approach on a toxic dataset of the Portuguese language, outperforming several graph-based methods and achieving competitive results compared to transformer architectures.

1 Introduction

Toxic comments, posts, and other types of content became more common in social media nowadays. They contain forms of non-acceptable language (profanity), which may be concealed or explicit, including insults and threats directed to a group or individual (Zampieri et al., 2019). These comments spread rapidly on the internet, especially on social networks where they find acceptance, and may culminate in several threats to individuals, becoming a serious concern for government organizations, online communities, and social media platforms.

The term toxic comment is commonly found in literature as harmful speech, hate speech, or offensive language. Toxic comment may be viewed as negative online behaviors, i.e., comments that are rude, disrespectful, may contain hate speech, or otherwise likely to make someone leave a discussion1. Schmidt and Wiegand (2017) define hate speech as any communication that disparages a person or a group based on some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics. Also, it may occur with different linguistic styles, even in subtle forms or when humour is used (Fortuna and Nunes, 2018). It is important to highlight that fighting these types of comments is of utmost importance since they are a crime in several countries.

To deal with toxic comments, most approaches adopt supervised-machine learning techniques and are mainly focused on the English language (Poletto et al., 2020). These approaches range from surface-level features, as Bag-Of-Words (Paiva et al., 2019), linguistics features, as Part-Of-Speech information (Chen et al., 2012), deep neural networks, as Long Short-Term Memory (LSTM) (Fortuna et al., 2019) and Convolutional Neural Networks (CNN) (Badjatiya et al., 2017) to Transformer architectures (Leite et al., 2020). Despite interesting results achieved by Transformer architectures, there are still several rooms to be explored in this research area.

In this paper, we developed a semi-supervised strategy to detect toxic comments in the Brazilian Portuguese language. Semi-supervision is the problem of learning from labeled and unlabeled data (Abney, 2007; Subramanya and Talukdar, 2014), in which given a point set $X = \{x_1, \ldots, x_l, x_{l+1}, \ldots, x_n\}$ and a label set $L = \{1, \ldots, c\}$, the first $l$ points have labels $\{y_1, \ldots, y_l\} \in L$ and the remaining points are unlabeled (Zhou et al., 2004).

We modeled that problem as a heterogeneous network. The structure of our graph was inspired by de Sousa et al. (2020) and Anchiêta et al. (2020). These authors modeled the tasks of helpfulness prediction and paraphrase identification as a heterogeneous network, respectively. For that, they defined an undirected unweighted graph with two node types: sentence and token. However, we have

1https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview
created a weighted graph based on pre-trained word embeddings. The weight between sentence and token nodes is the average of the embedding values for that token. Figure 1 depicts an example of a sentence modeled as a graph. From this figure, we may see two node types: token and sentence, and an undirected and weighted edges between the sentence and tokens nodes.

Figure 1: Example of a graph model for the sentence “Holy shit, I miss playing midnight club”.

To extract features from the graph structure, we used a regularization algorithm that propagates labels from a small set of labeled nodes to the entire graph.

We evaluated the approach using the ToLD-Br corpus (Leite et al., 2020). It has twenty-one thousand annotated tweets as either toxic or non-toxic language. Also, we compared our strategy with different graph-based methods and with transformer-based methods. Our method outperformed all graph-based approaches and achieved competitive results compared to transformer-based methods, using only 10% of labeled nodes.

The reminder of this paper is structured as follows: Section 2 briefly presents related work. In Section 3, we show the used corpora. Section 4 details our developed approach. In Section 5, we analyze the conducted experiments. Finally, Section 6 concludes the paper, presenting future directions.

2 Related Word

As aforementioned, the main approaches to detect toxic comments are based on supervised machine learning. Here, we briefly present the main works.

Most of the works that study this task commonly point first to surface-level features, such as bag of words and lexicon-based approaches, with negative words as features (Gitari et al., 2015; Waseem and Hovy, 2016; Waseem et al., 2017; Schmidt and Wiegand, 2017).

More recently, neural networks-based strategies and transformer-based architectures has been applied to hate speech detection due to the good results achieved in various tasks. Banerjee et al. (2020) evaluated pre-trained word embeddings with CNN networks to hate speech detection for the Indian language. Rizwan et al. (2020) explored transfer-learning of embeddings models to Roman Urdu and developed a CNN-gram network to hate speech classification for that language. Duwairi et al. (2021) investigated the ability of CNN, CNN-LSTM, and BiLSTM-CNN to classify hate speech in Arabic. Plaza-del Arco et al. (2021) compared two pre-trained language models, such as BERT (Devlin et al., 2019) and XLM (Conneau and Lample, 2019) trained to detect hate speech in the Spanish language.

For the Portuguese language, most of the works follow the trend of supervised approaches. de Pelle and Moreira (2017) created a dataset consist of 1,250 offensive comments and developed a baseline method based on n-gram features to classify offensive comments in their dataset. Fortuna et al. (2019) created a hate speech dataset composed of 5,668 tweets and developed a baseline classification using pre-trained word embeddings and LSTM in their dataset. Coutinho and Malheiros (2020) trained a logistic regression using superficial features for sentiment analysis. Then, they evaluated that model into a homophobia corpus to detect homophobic posts.

Although there are some efforts to detect non-acceptable language in Portuguese, they evaluate the developed approach in their own corpus, making a fair comparison among the models difficult. Moreover, these corpora are much smaller when compared to corpora of other languages (Poletto et al., 2020) and than the ToLD-Br corpus. This fact makes the development of robust strategies to handle toxic comments difficult, as they usually require a large corpus.

3 ToLD-Br Corpus

Toxic Language Dataset for Brazilian Portuguese (ToLD-Br) (Leite et al., 2020) is a very recent dataset with Twitter posts in the Brazilian Portuguese language. It has 21K tweets manually annotated into seven categories: non-toxic, LGBTQ+phobia, obscene, insult, racism, misogyny, and xenophobia. The corpus is the largest dataset available for toxic data analysis in social
media for Portuguese and the first dataset with demographic information about annotators.

Besides seven categories, the authors released a binary version of the corpus for the binary classification task, as shown in Table 1.

<table>
<thead>
<tr>
<th>Label</th>
<th>Train.</th>
<th>Valid.</th>
<th>Test</th>
<th>Prop.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toxic</td>
<td>7,375</td>
<td>908</td>
<td>972</td>
<td>44%</td>
</tr>
<tr>
<td>Non-toxic</td>
<td>9,425</td>
<td>1,192</td>
<td>1,128</td>
<td>56%</td>
</tr>
</tbody>
</table>

Table 1: Binary version of the ToLD-Br corpus.

As one can see in Table 1, the corpus has a little more non-toxic than toxic tweets. In this paper, we adopted the binary version of the corpus, i.e., our objective is to identify if a comment is toxic or non-toxic.

In what follows, we detail our strategy to handle toxic texts.

## 4 Semi-supervised approach

We organized the strategy into four steps, as illustrated in Figure 2. Subsections 4.1, 4.2, 4.3, and 4.4 describe the stages.

### 4.1 Pre-processing

In the pre-processing, we normalized and cleaned the tweets. In the first one, we applied the Enelvo tool (Costa Bertaglia and Volpe Nunes, 2016) to normalise abbreviated and repeated words. In the second one, we simply clean URLs, emojis, and tweet mentions.

### 4.2 Graph-Based Method

We modeled toxic comments detection as a heterogeneous network since this network type contains abundant information with structural relations (edges) among multi-typed nodes as well as unstructured content associated with each node (Zhang et al., 2019). Graph structures have been used for several tasks, such as: topic model, name disambiguation, scientific impact measurement, and others, obtaining good results (King et al., 2014).

We defined a undirected and weighted graph as $G = (V, E, W)$, where $V$ is a set of vertices $V = \{v_1, ..., v_n\}$, $E$ indicates a set of edges $E = \{e_1, ..., e_n\}$, and $W$ is a weighted adjacency matrix, in which $W_{i,j}$ denotes the weight of an edge between nodes $i$ and $j$. We defined two node types: token and sentence and two constraints not allowing link among tokens nodes or among sentences nodes.

The strategy of weighting links between a token and a sentence node is straightforward. The weight is the average of embedding vectors of the token node. To get embedding values for each token, we used 100-dimensional GloVe embeddings for the Portuguese language (Hartmann et al., 2017). Figure 3 shows the scheme of the network designed for this task.

![Figure 3: The network scheme for weighted edges.](image)

One can see that the edges are undirected and weighted, and a sentence node may share several token nodes whenever the token is in the sentence, i.e., the edges between token nodes and sentence nodes are based on word occurrence in sentence.

### 4.3 Regularization

To extract the features regarding the network object classes, we applied a regularization method to the graph. Regularization is a kind of semi-supervised (or transductive) classification method that aims to find a set of labels, minimizing a cost function and satisfying two conditions: (i) the method needs to be consistent with the set of labels manually annotated and (ii) the method needs to be consistent with the network topology, considering that nearest neighbors tend to have the same labels (Ji et al., 2010).

We used the learning with Local and Global Consistence (LGC) (Zhou et al., 2004) as a regularization method. The algorithm designs a classi-
fying function that is sufficiently smooth concerning the intrinsic structure collectively revealed by known labeled and unlabeled points. Thus, the LGC lets every point iteratively spread its label information to its neighbors until a global stable state is achieved (Gui et al., 2014). Also, it allows the class information of the labeled objects to be changed during the classification as objects may be erroneously labeled and, consequently, decrease the performance of the classification. More than that, the algorithm diminished the influence of objects with a high degree (many neighboring objects), therefore, these objects do not have excessive influence in the classification.

To execute the algorithm, a set of nodes need to be pre-labeled. The regularizer randomly pre-labeled, i.e., supposing that the percentage of pre-labeled nodes is equals 5%, it means that 0.25% of each class is randomly pre-labeled. As a result, the regularizer produces values related to coordinates for each object in the network, as shown in Table 2.

<table>
<thead>
<tr>
<th>Id</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.004567</td>
<td>0.001456</td>
<td>1</td>
</tr>
<tr>
<td>255</td>
<td>0.002789</td>
<td>0.008763</td>
<td>0</td>
</tr>
<tr>
<td>878</td>
<td>0.001998</td>
<td>0.005342</td>
<td>0</td>
</tr>
<tr>
<td>233</td>
<td>0.008764</td>
<td>0.003215</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2: Example of regularizer output.

From Table 2, **Id** is the object identifier, **Values** refer to coordinates of each object in the network, and **Label 1** shows toxic, while **Label 0** is a non-toxic tweet.

4.4 Classification

With the regularization values, we fed several machine learning algorithms to identify and predict toxic comments. We experimented Multi Layer Perceptron, Naïve Bayes, Decision Tree, Support Vector Machine, and Gradient Boosting from the Scikit-Learn library (Pedregosa et al., 2011).

In the following section, we detailed our carried out experiments, then, the achieved results are presented.

5 Experiments and Results

In order to produce coordinate values for each object from the regularizer, we ranged the number of pre-labeled nodes from 5% to 30%. Then, we applied the machine learning algorithms to train and classifier toxic comments.

We achieved the best result with the Gradient Boosting classifier using only 10% of the pre-labeled nodes i.e., the classification does not improve after this percentage. Table 3 shows the achieved results. It is important to say that only the training set is pre-labeled.

<table>
<thead>
<tr>
<th>Pre-labeled (%)</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Toxic</td>
</tr>
<tr>
<td>5</td>
<td>0.73</td>
</tr>
<tr>
<td>10</td>
<td>0.73</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 3: Achieved results with the gradient boosting classifier.

Besides our approach, we evaluated other graph models of different structures. First, we used the network graph developed by Anchiêta et al. (2020). That graph does not use weight between the nodes. Second, we used the Term Frequency-Inverse Document Frequency (TF-IDF) as weight instead of the average of embeddings. Third, we used bigrams and trigrams as nodes rather than token nodes. Finally, we used the Pointwise Mutual Information (PMI) measure (Church and Hanks, 1990) as the weight between the bi and trigrams nodes. For these approaches, we adopted the same regularization algorithm, ranging the pre-labeled nodes from 5% to 30%. In Table 4, we present the best-achieved results.

<table>
<thead>
<tr>
<th>Pre-labeled</th>
<th>Method</th>
<th>F-score</th>
<th>Classifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>30%</td>
<td>Trigrams without weight</td>
<td>0.70</td>
<td>MLP</td>
</tr>
<tr>
<td>30%</td>
<td>Bigrams without weight</td>
<td>0.72</td>
<td>MLP</td>
</tr>
<tr>
<td>30%</td>
<td>Trigrams + PMI</td>
<td>0.69</td>
<td>GB</td>
</tr>
<tr>
<td>30%</td>
<td>Bigrams + PMI</td>
<td>0.62</td>
<td>GB</td>
</tr>
<tr>
<td>30%</td>
<td>Unigrams + TF-IDF</td>
<td>0.69</td>
<td>GB</td>
</tr>
<tr>
<td>30%</td>
<td>Anchiêta et al. (2020)</td>
<td>0.68</td>
<td>MLP</td>
</tr>
</tbody>
</table>

Table 4: Comparison among graph-based approaches.

From this table, our graph modeling and the gradient boosting classifier achieved better results than these other graphs, as well as classifier variations. This, we think, is because of the embedding value among the graph nodes since it is able to capture morphological, syntactic, and semantic knowledge of a word. As we used the average word embedding value, it includes information from all of the
individual vector values, working as an overall summary of all vector values.

We further compared our strategy with other graph-based approaches: Text Graph Convolutional Network (TextGCN) (Yao et al., 2019) and Heterogeneous Graph Attention Network (HGAT) (Yang et al., 2021). The former models the whole text corpus as a document-word graph with word co-occurrence relations and applies GCN for classification. The latter models the texts using a heterogeneous information network framework and adopts heterogeneous graph attention to embed that framework for text classification based on a dual-level attention mechanism. Finally, we compared our approach with a transformer-based method as it has achieved remarkable results in several areas of Natural Language Processing (NLP). We compared our strategy with BR-BERT (Leite et al., 2020), which is a monolingual BERT, and M-BERT (Leite et al., 2020), which is a multilingual BERT. Table 5 shows the comparison between these methods.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Model</th>
<th>F-score</th>
<th>Macro F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Toxic</td>
<td>Non-toxic</td>
</tr>
<tr>
<td>Graph</td>
<td>TextGCN</td>
<td>0.70</td>
<td>0.69</td>
</tr>
<tr>
<td></td>
<td>HGAT</td>
<td>0.55</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>BR-BERT</td>
<td>0.79</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>M-BERT</td>
<td>0.77</td>
<td>0.75</td>
</tr>
<tr>
<td>Graph</td>
<td>Ours</td>
<td>0.73</td>
<td>0.74</td>
</tr>
</tbody>
</table>

Table 5: Comparison between graph-based and transformer-based methods with our strategy.

As we can see from Table 5, our approach outperformed the graph-based methods and reached a competitive result compared to transformer models. Although our strategy did not outperform transformers, we believe the results are very promising, since it requires much less computational power than transformers. Moreover, our method requires less annotated data (only 10%) than transformers to achieve interesting results.

Our approach is available at https://github.com/rafaelanchieta/toxic.

6 Conclusion and Future Work

In this paper, we explored a semi-supervised strategy to deal with toxic comments from Twitter. We modeled the texts as a heterogeneous network graph with two node types and weighted edges among nodes. Then, we applied a regularization algorithm to extract features related to the toxic texts. Finally, we used these features to feed a classifier to identify and predict toxic comments. Our approach outperformed several graph-based methods and achieved a competitive result compared to the BERT model, using only 10% of the corpus. We hope that this graph model brings insights to hate speech detection research, helping to improve the results. Furthermore, our strategy may be employed in other languages, as it only requires an embedding representation.

As future work, we intend to explore the graph structure, analyzing some network measures, such as degree, centrality, community identification, and others. Also, we aim to examine contextual embeddings rather than traditional embeddings.

Acknowledgments

The authors are grateful to CAPES for supporting this work.

References


Ming Ji, Yizhou Sun, Marina Danilevskiy, Jiawei Han, and Jing Gao. 2010. *Graph regularized transductive classification on heterogeneous information networks*. In *In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 570–586, Barcelona, Spain. Springer.


Graph-based Argument Quality Assessment

Ekaterina Saveleva and Volha Petukhova and Marius Mosbach and Dietrich Klakow
Spoken Language Systems Group, Saarland Informatics Campus
Saarland University, Saarbrücken, Germany
{esaveleva,v.petukhova,mmosbach,dietrich.klakow}@lsv.uni-saarland.de

Abstract

The paper presents a novel discourse-based approach to argument quality assessment defined as a graph classification task, where the depth of reasoning (argumentation) is evident from the number and type of detected discourse units and relations between them. We successfully applied state-of-the-art discourse parsers and machine learning models to reconstruct argument graphs with the identified and classified discourse units as nodes and relations between them as edges. Then Graph Neural Networks were trained to predict the argument quality assessing its acceptability, relevance, sufficiency and overall cogency. The obtained accuracy ranges from 74.5% to 85.0% and indicates that discourse-based argument structures reflect qualitative properties of natural language arguments. The results open many interesting prospects for future research in the field of argumentation mining.

1 Introduction

Argumentation modelling and mining are steadily gaining attention of the broad natural language processing and engineering community. In many studies and applications, assessment of the argument quality plays an important role. The ability to construct good arguments and engage in argumentative discussions is assessed by argumentation systems focusing on training hypothetical reasoning, creating and structuring arguments (Ashley et al., 2007), preventing opinion manipulation, detecting inconsistent arguments in online discussions and addressing different standpoints, attacking or supporting claims with evidence (DebateGraph1 and TruthMapping2) as well as on the use of multimodal rhetorical devices (Petukhova et al., 2017a). Assessment of argument quality, its organization, clarity, adherence and strength, are approached by several authors as sub-tasks in the evaluation of written essays (Stab and Gurevych, 2014; Persing and Ng, 2015; Wachsmuth et al., 2016; Stab and Gurevych, 2017). Online content is searched to filter or weight the validity of statements and factoids (Rowe and Butters, 2009), to identify fake news and false claims (Popat et al., 2018) and to detect opinion manipulation (Cambria et al., 2010). While the acceptability of an argument in the presence of other supporting or attacking arguments has been addressed (Dung, 1995; Cayrol and Lagasquie-Schiex, 2005), ‘local’ argument quality still deserves our attention – an argument built on a certain set of conditions, is logically strong, rhetorically convincing, socially undistorted by virtue of its intrinsic properties.

In this paper, we present a novel approach to assessing the structural strength and inferential weakness of arguments as merits of argument cogency. The approach relies on the discourse-based reconstruction of argumentation schemes. For this, we apply state-of-the-art discourse parsers and machine learning models to reconstruct argument graphs where the identified discourse units are represented as nodes and the classified discourse relations between them as edges. A Graph Neural Network (GNN) model is built to predict the quality (low vs high) of the reconstructed argument graphs in terms of argument acceptability, relevance, sufficiency and overall cogency.

The paper is structured as follows. Section 2 defines the conceptual framework within which the study is performed. We provide the definition of an argument and elaborate on its internal structure. In Section 3, we survey related work on argument quality assessment. Section 4 presents the argument graph reconstruction approach. The performed GNN-based quality assessment experiments are discussed and results are reported in...
Section 5. Section 6 summarises our findings and outlines directions for future research.

2 Argument and Its Structure

An argument may be considered as an atomic entity without an internal structure. For instance, an argument is defined as an overall position held by a person towards an idea or attitude, e.g. a stance in ‘favour’ or ‘against’ a certain motion (Somassundaran and Wiebe, 2009). A structured argumentation model is an essential element for the tasks aiming at understanding and emulating of human inference, investigating patterns of reasoning, focusing at extraction and validity assessment of arguments. A simple argument structure is then defined as consisting of a claim that is supported by evidence(s) (Mochales and Moens, 2011; Aharoni et al., 2014). A claim is an assertion that an argument aims to prove, i.e. a claim is a conclusion whose merit must be established. Evidence comprises propositions which give reasons or grounds for drawing the conclusion.

This general argument definition has been translated into several discourse-based schemes for analysing and evaluating natural language arguments (Teufel, 1999; Palau and Moens, 2009). An argument is modelled as a group of Argumentative Discourse Units (ADUs) – text segments corresponding to propositions that are argumentatively relevant and have their own argumentative function (Peldszus and Stede, 2013). An EDU can function as a claim, as an evidence or as a conclusion. An ADU can be identified as a collection of several Elementary Discourse Units (EDUs) which correspond to clauses in written discourse and to dialogue acts in spoken discourse (Petukhova et al., 2016). Discourse relations such as Justification, Motivation, Cause, and Exemplification can be used to identify how propositions are related to each other, inferring the type of support that is expressed. A claim may be summarized or re-stated in a conclusion. Figure 1 depicts a general discourse-based argument structure.

ADUs reflect different ways to provide support for a claim, i.e. links between them express the level of support that evidence provides to the claim and the level of their sufficiency to draw a valid conclusion. Figure 2 provides an example of an argument. EDUs (solid-line rectangles) are combined by means of discourse relations into ADUs (dotted-line rectangles) which are connected to each other by support links. Evidence may either together (linked support, e.g. Evidence 2.1 and 2.2) or independently (multiple support, e.g. Evidence 1, 2 and 4) support a conclusion. A premise may provide support for another premise and indirectly support a conclusion (serial support, e.g. Evidence 3 and 2). A special form of lending support to a claim is that of providing examples (example support, e.g. Evidences 4.1 and 4.2).²

3 Related Work

Clear properties of a good argument and successful argumentation are not easy to define. Wachsmuth et al. (2017) proposed a unified taxonomy of argumentation quality assessment that resulted from an extensive analysis of numerous existing approaches. The assessment comprises three quality dimensions: cogency, effectiveness and reasonableness. Argument quality assessment aims at answering the question how logical, persuasive or convincing the given argument is, and how rhetorically appealing it is for the targeted audience.

Evaluation of argument cogency is based on the truthfulness and logical coherence of arguments. An argument is cogent if it has acceptable premises that are relevant and sufficient to support the conclusion (Johnson and Blair, 2006; Govier, 2013). A premise is acceptable if it is rationally worthy of being believed to be true (Wachsmuth et al., 2017). According to Govier (2013), a premise is locally acceptable if it is supported by a cogent sub-argument or another cogent argument; it is a matter of common knowledge, testimony or expert view (appeal to authority). A statement A is positively relevant to another statement B if and only if the truth of A counts in favour of the truth of B. This means that

²For a discussion on other types of support links we refer to Palau and Moens (2009) and Peldszus and Stede (2013).
According to the National Association for PET Container Resources, PET water containers are now the most recycled container in curb side program by weight and by number! Evidence 1

More than a million in the United States purchase bottled water every day which is helping the economy to exit the current recession.

Evidence 2

Bottled water also only makes less than 1% the world’s waste and 1% is a trivial proportion.

Evidence 3

Claim/conclusion

Banning plastic bottles would be a huge mistake in this very moment

Evidence 4

In addition, bottled water is a safer alternative as compared to tap water in some countries.

Evidence 2

2.1

Cause Exemplification

2.2

Elaboration

Figure 2: Argument example from Dagstuhl-15512 ArgQuality Corpus (Wachsmuth et al., 2017) annotated with core ISO 24617-8 core discourse relations (Bunt and Prasad, 2016) and support links observed.

A provides some evidence for B, or some reason to believe that B is true. An argument is locally sufficient if all premises together provide sufficient reasons to accept the conclusion. The preconditions of the argument sufficiency are rooted in its local acceptability and its local relevance (Govier, 2013). The local sufficiency of an argument is often called inferential sufficiency and it holds if one of the following logical patterns is applicable: deductive entailment, conductive support, inductive support and analogy.

Argument quality is found to correlate with the argument’s actual persuasive success. Persuasion is defined as a process of encouraging people to do or believe something through argument. Here, many factors are relevant, including psychological effects of argument memorisation, replication and reviewing (Kumkale and Albarracín, 2004). Certain argumentation patterns are acknowledged as more persuasive than others, however they may differ in different domains. Hornikx (2008) experimentally investigated lay people’s expectations about the persuasiveness of anecdotal, statistical, causal, and expert evidence, and compared these expectations with the actual persuasiveness of these evidence types. Van Eemeren and Grootendorst (2004) defined symptomatic (sign), comparison (resemblance) and causal (consequence) argumentation, and specified what argumentative patterns are more suitable/persuasive for what communicative types in various domains. For persuasive essays, different quality dimensions of argumentation were studied such as essay’s organization (Persing et al., 2010), thesis clarity (Persing and Ng, 2013), prompt adherence (Persing and Ng, 2014) and argument strength (Persing and Ng, 2015). These studies exploit a complex feature-rich approach to predict a score for each essay based on its content or style along with all of these categories. The study of Persing and Ng (2017) looks at the argument persuasiveness from a different point of view: it does not try to estimate how persuasive an argument is but attempts to explain why an argument is experienced as unpersuasive. Research has also targeted various interactive aspects, e.g. capturing the interactions between participants on argument level (Ji et al., 2018) and providing feedback regarding the argument persuasiveness (Ke et al., 2018).

Many studies explore the aspect of argument convincingness4. In contrast to cogency, which is based on the truthfulness and logical coherence of arguments, convincingness is related to subjective perception by the audience (Wei et al., 2016). Experiments were performed to detect more convincing arguments (Habernal and Gurevych, 2016; Simpson and Gurevych, 2018) and evidence (Gleize et al., 2019).

The rhetorical force of an argument should not be underestimated. Due to the use of powerful rhetorical devices, even a not very cogent argument may be perceived as convincing (Petukhova et al., 2017b; Hirschberg, 2002). People generally associate certain speech, personality and interaction features with what they think is a persuasive argument.

4It should be noted here that persuasiveness and convincingness of an argument are terms that are often used interchangeably.
More broadly, the persuasion literature of the last decades has shown that an argument that has higher perceived competence (e.g., evidence-based expert knowledge) and/or higher warmth (e.g., more likeable and trustworthy) is more convincing (Petty and Cacioppo, 1986; Albarracin et al., 2019).

The study of Wachsmuth et al. (2016) suggests an argument quality assessment approach that focuses solely on the argument structure, and defines statistical patterns in the structure of essays and define novel features that are evaluated in argumentation-related essay scoring tasks. The present study investigates the structural properties of a cogent argument and assesses its inferential strength, i.e. structures of inference (argumentation schemes) predicted from the associated amount, depth and type of evidence provided to the claim.

4 Argument Graph Reconstruction

We define argument graph reconstruction to involve: (1) segmentation of a text into EDUs; (2) discourse relation detection and classification between them; (3) identification and classification of ADUs based on the classified discourse relations; and (4) argument completion, i.e. reconstruction of the implicit units to achieve a complete argument structure, see Figure 3.

We performed joint and two-stage segmentation and classification of EDUs. For the joint segmentation and classification, the full PDTB parser developed by Lin et al. (2010) was applied. We observed that the parser failed to identify many EDU spans. Rather low overall F1 scores of 21.20% for exact segment boundaries match and 5-class discourse relation classification were achieved on the Penn Discourse Tree Bank 1.0 corpus (PDTB 1.0, Prasad et al. (2005)), see the right part of Table 1. However, we observed, that in case of the correct span identification, relations classification was reasonably accurate. Misclassified cases mostly belonged to the implicit discourse relations as they were more difficult to classify then the explicit ones, a well known problem reported in the literature.

The two-stage segmentation and classification was performed applying the BiLSTM-CRF based segmentation model NeuralEDUSeg developed by Wang et al. (2018) and the XLNet-large discourse relations classification model by Yang et al. (2019). A segmentation performance of 68.55% in terms of F1 score was achieved when testing on the PDTB 1.0 and PDTB 2.0 datasets (PDTB 2.0, Prasad et al. (2008)). For discourse relation recognition with the XLNet model designed by Yang et al. (2019), we first carried out a binary classification to establish whether any relation exists between the identified units, i.e. discriminate between the Rel class which includes any type of discourse relations and NoRel types. The former comprises explicitly marked (Explicit), implicitly marked (Implicit) and alternatively lexicalized (AltLex) discourse relations, the later includes EntRel relation which is not a discourse relation between clauses but an entity-based coherence relation. Subsequently, we performed five-class top-level (L1) and ten-class fine-grained (L2) relations classification. Table 4 in Appendix I provides an overview of the PDTB discourse relation and their distribution in the PDTB 1.0 and the newer PDTB 2.0 corpora.

Since class distributions were unbalanced in all classification settings, re-sampling was performed: up-sampling of the under-represented NoRel class in binary classification by adding synthetic samples. For this random EDUs from different textual units were combined. In addition to this, down-sampling of the majority classes in the multi-class settings was performed. For the training and evaluation procedures, we fine-tuned each encoder model following the suggestions of Mosbach et al. (2021) and trained it for 10 epochs using a learning rate of

---

5The same observation was made by Hewett et al. (2019).
Table 1: Performance overview on the joined EDU segmentation and 5-class discourse relation classification task with Lin et al. (2010) parser in terms of F1 scores (in %) on the PDTB 1.0 corpus (left); and on the two-stage segmentation and classification tasks performing EDU segmentation with the NeuralEDUSeg model (Wang et al., 2018) in terms of F1 scores (in %) on the PDTB 2.0 corpus, and 5- and 10-class discourse relation classification with the fine-tuned XLNet-large model (Yang et al., 2019) in terms of accuracy (in %) on the DagStuhl corpus (right).

We observed that some argument components, often claims, are implicit, see also Wachsmuth et al. (2017). Without the claim or conclusion, an argument structure is incomplete. Therefore, we reconstructed a claim for every topic in the corpus, either ‘for’ or ‘against’ stance it may present. The reconstructed claim is a simple sentence corresponding to a single EDU, see Table 5 in Appendix II for selected examples.

The identified Dagstuhl arguments are of different length and have various, often complex discourse-based structures distinguishable through diverse linking patterns and number of evidences provided by an arguer to support a claim. Figure 4 provides an example of the identified discourse-based argumentation scheme. The upper node represents the claim *Books are better than TV* which was supported by seven evidence statements, six of them connected to the claim by means of *Contingency.Cause* relation and one by *Expansion.Instantiation*. Five of the evidence statements correspond to one EDU, whereas the other two are more complex and consist of two EDUs.

Finally, argument structures were represented as graphs where the detected EDUs spans are represented as nodes and the classified discourse relations – as edges visualising number and level of supporting evidence through links to the claim and premises. 2,278 reconstructed arguments have 303 structures (argumentative schemes) specifying 172 unique reasoning patterns. Figure 6 in Appendix III provides the most frequent examples of the reconstructed discourse-based argument structures.

5 Argument Quality Assessment Using Graph Neural Networks

Our main assumption is that arguments constructed to follow certain patterns and containing particular discourse relations are of higher quality, i.e. inferentially stronger, than others. Thus, the amount and type of evidence matter. For example, a widely used structure of debate arguments is known as the ARE. ARE comprises a claim of an Argument supported by a Reason and an Evidence, see also Petukhova et al. (2016). Another commonly used argument structuring technique is called chunking (Johnson, 2009). Here, arguers generalise from a claim (chunking up), provide a specific example (chunking down) or draw analogies (chunking side-
Thus, an argument which contains *Cause* or *Instantiation* relations making a claim to be justified and explained, is expected to be of high quality.

A large variety of identified discourse relations may indicate that an argument is very elaborate. However, very lengthy arguments are often difficult to comprehend and may be rhetorically less persuasive.

Particular structures can be important to compute local sufficiency. For instance, conductive reasoning may be expressed by the number of first-level evidence supporting the main claim independently (multiple support) and together (linked support) connected by means of *Expansion* relations. For inductive support, Exemplification discourse relations can be analysed as evidence providing an example support. Finally, the argumentation depth can be relevant for the argument sufficiency assessment and can be computed by looking at evidence which is linked to other evidence statements providing a serial support. Obviously, not only linking patterns but also evidence content would impact the argument quality.

Arguments in the Dagstuhl15512 ArgQuality corpus were annotated by seven independent annotators across 15 quality dimensions including four for argument cogency: acceptability, relevance, sufficiency and overall cogency. Quality scores from 1 (low) to 3 (high) were assigned. A fair inter-annotator agreement for all cogency dimensions was reached ranging from .44 to .47 in terms of Krippendorff’s α (Wachsmuth et al., 2017). Distribution of the annotated quality classes resulted in a rather unbalanced training set, in particular for the sufficiency dimension, therefore we combined the minority class with the adjacent one defining a binary classification task predicting arguments of a *lower* and of a *higher* quality, see Table 2 for distributions.

To assess the argument cogency, we employed a Graph Neural Network (GNN) model which is able to generalize over manifold structures. Errica et al. (2019) presents an overview of GNNs models for graph classification, e.g. DGCNN, DiffPool, ECC, GIN, GraphSAGE. However, none of these models exploit edge features required for our application so it incorporates discourse relation information.

### 5.1 Architecture Overview

For our experiments, we use the Graph Attention Network (GAT) model by Veličković et al. (2018). Initial inputs to the model include the node feature matrix $X^0$ as presented in Figure 5 (left). The matrix is fed to the GNN layer which is able to handle binary edge features, i.e. in our case the model can only handle the existence or absence of a relation between two EDUs. The single-head and multi-head attention mechanisms used within the GNN layer are illustrated on Figure 5 (center and right). As a result, a new node matrix $X^1$ is
produced. The procedure is repeated for every subsequent layer. Our network consists of three such layers. For graph classification, an average pooling layer is applied to the first dimension of $X^{L}$, i.e. the feature matrix is reduced to a single vector representing the whole graph. Subsequently, the fully connected layer is applied to the vector whose outputs are used as logits for the final classification.

The model described above allows incorporating various features in the node feature matrix $X^{0}$. We experimented with three different settings. In the first design, no node features were available. Hence, that model predicts argument quality based solely on the graph structure. In the second design, we encoded the texts corresponding to each node using GloVe embeddings taking the mean of all word embeddings of an EDU to obtain the EDU representation. In the third setting, we used BERT embeddings of an EDU as node features using the DistilBERT pre-trained model optimized for Semantic Textual Similarity task (Reimers and Gurevych, 2019). The vector dimensionality for each of the settings described above was 300, 300 and 768 dimensions, respectively.

### 5.2 Experimental Results

We trained a model for each quality dimension (cogency, acceptability, relevance, sufficiency) varying the number of epochs from 100 to 1000. Subsequently, we determined the best training setting based on the validation set accuracy. We observed that the performance on quality dimensions applying different either GloVe or BERT node features depends on the number of training epochs. Furthermore, we learned that the models with the GloVe and BERT node features reduced loss much faster than the models with uniform node features which indicates a possible overfitting. Therefore, we experimented with dropout rates from 0.2 to 0.6 applying it to the output of the last hidden layer. Test accuracy of BERT-based models improved for all dimensions except for acceptability. 10 training runs for each quality dimension showed that the test accuracy for certain models did not vary much with different dropout rates, the results were also proven not to be statistically significant according to the paired t-test. Table 3 reports the highest mean test accuracy achieved across five runs.

All trained quality assessment models outperformed their corresponding baselines indicating that the graph structure and node features incorporate useful information. Models generally were proven to be resistant to class imbalance. Resampling or weighted training and testing did not result in better performance, which indicates that more sophisticated methods are required to improve results. Further findings suggest that there is no ‘universal’ training setting for various quality dimensions: to achieve acceptable performance some models require longer training and different sets

---

Table 2: Class distribution in the training, validation and test sets in the Dagstuhl15512 ArgQuality corpus.

<table>
<thead>
<tr>
<th>class</th>
<th>overall</th>
<th>training set</th>
<th>validation set</th>
<th>test set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lower</td>
<td>higher</td>
<td>lower</td>
<td>higher</td>
</tr>
<tr>
<td>cogency</td>
<td>143 (47.2%)</td>
<td>160 (52.8%)</td>
<td>128 (52.9%)</td>
<td>14 (46.7%)</td>
</tr>
<tr>
<td>acceptability</td>
<td>71 (23.5%)</td>
<td>232 (76.5%)</td>
<td>185 (76.4%)</td>
<td>8 (26.7%)</td>
</tr>
<tr>
<td>relevance</td>
<td>183 (60.4%)</td>
<td>120 (39.6%)</td>
<td>96 (39.7%)</td>
<td>20 (66.7%)</td>
</tr>
<tr>
<td>sufficiency</td>
<td>167 (55.1%)</td>
<td>136 (44.9%)</td>
<td>133 (55%)</td>
<td>109 (45%)</td>
</tr>
</tbody>
</table>

Figure 5: **Left**: Our model architecture. Tensor $X$ is a matrix representation of the node features of the graph with dimensions $N \times F$ where $N$ is the number of nodes and $F$ is the number of features (Gong and Cheng, 2019). **Center**: The attention mechanism employed by GAT layer (Veličković et al., 2018). **Right**: an illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighbourhood (Veličković et al., 2018).
Table 3: Classification accuracies on Dagstuhl15512 ArgQuality argument quality assessment applying a graph classification model. The results are reported for each quality dimension (cogency, acceptability, relevance, sufficiency) and using different node features (uniform, GloVe, BERT).

<table>
<thead>
<tr>
<th>quality dimension</th>
<th>majority classifier</th>
<th>uniform features</th>
<th>GloVe features</th>
<th>BERT features</th>
</tr>
</thead>
<tbody>
<tr>
<td>cogency accuracy</td>
<td>54.8</td>
<td>72.5</td>
<td>68.0</td>
<td>63.0</td>
</tr>
<tr>
<td>epochs</td>
<td>400</td>
<td>800</td>
<td>800</td>
<td></td>
</tr>
<tr>
<td>dropout</td>
<td>-</td>
<td>0.2</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>acceptability</td>
<td>accuracy 80.6</td>
<td>85.0</td>
<td>84.5</td>
<td>85.0</td>
</tr>
<tr>
<td>epochs</td>
<td>300</td>
<td>600</td>
<td>600</td>
<td></td>
</tr>
<tr>
<td>dropout</td>
<td>-</td>
<td>0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>relevance</td>
<td>accuracy 54.8</td>
<td>69.5</td>
<td>74.5</td>
<td>74.0</td>
</tr>
<tr>
<td>epochs</td>
<td>300</td>
<td>200</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>dropout</td>
<td>-</td>
<td>0.2</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>sufficiency</td>
<td>accuracy 54.8</td>
<td>64.0</td>
<td>74.5</td>
<td>69.0</td>
</tr>
<tr>
<td>epochs</td>
<td>500</td>
<td>200</td>
<td>900</td>
<td></td>
</tr>
<tr>
<td>dropout</td>
<td>-</td>
<td>0.4</td>
<td>0.6</td>
<td></td>
</tr>
</tbody>
</table>

of node features. Surprisingly, sophisticated node features do not always lead to a better model performance. For instance, for the cogency dimension the model without node features significantly outperformed the GloVe- and BERT-based models which may suggest that the argument structure alone is sufficient to accurately predict argument cogency. This confirms our initial assumption. However, we are cautious with this conclusion and emphasise that further experiments on larger datasets are needed. Training on such small training (242 instances), validation (30 instances) and test (30 instances) sets is a challenging task causing oscillation in the validation and test accuracies. Model instability was also caused, in our view, by a large variety of graph structures which can possibly be resolved by graph pruning or graph unification.

6 Conclusions and Future Work

We presented an approach to the assessment of argument quality, in particular its cogency, evaluating the structural strength of the argumentation schemes applied by an arguer. Argumentation schemes were represented as graphs reconstructed by applying the NeuralEDUSeg model developed by Wang et al. (2018) to segment a text into elementary discourse units and the fine-tuned XLNet-large model of Yang et al. (2019) to classify discourse relations between the identified units. Both segmentation and classification models showed reasonable performance in processing argumentative texts: F1 scores of 47.94% on the segmentation task were achieved; discourse relation classification accuracy ranges from 50.48% to 60.22% depending on the classification scenario (5- vs 10-class discourse relation classification). Parsed argumentative texts subsequently were used to reconstruct discourse-based argumentation structures as graphs of varying complexity reflecting reasoning patterns that emulate human inferencing. Given a graph structure, the argument acceptability, relevance, sufficiency and overall cogency were predicted. The trained models incorporated not only linking structures but also claim and evidence content as node features computed from GloVe and BERT embeddings. We tentatively concluded that overall argument cogency may be predicted based on the argument structure alone without computing sophisticated node text-based features. To ensure that this observation is not an artefact of our data, it needs to be tested on a larger argument set from various domains.

Limitations of the presented study call for further improvements. The next major step is to incorporate edge features which contain important information about the type of relations between claim and evidence units. Further, we intend to explore new discourse processing tools and experiment with mapping between different discourse analysis frameworks, e.g. Rhetorical Structure Theory (RST, Mann and Thompson (1988) and ISO 24617-8 DR-Core (ISO, 2016). Additional argumentative corpora will be explored as well as the assessment of the other quality dimensions.

References


Dolores Albarracín, Aasha Sunderrajan, Sophie Lohmann, Man-Pui Sally Chan, and Duo Jiang. 2019. The psychology of attitudes, motivation, and
persuasion. The handbook of attitudes, volume 1: Basic principles, pages 3–44.


Appendix I: PDTB discourse relation tagsets and corpus distribution

<table>
<thead>
<tr>
<th>Rel</th>
<th>L1 top-level relations</th>
<th>L2 fine-grained relations</th>
<th># Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoRel</td>
<td>NoRel</td>
<td></td>
<td>5464</td>
</tr>
<tr>
<td>Expansion</td>
<td></td>
<td>Conjunction</td>
<td>8763</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Restatement</td>
<td>3526</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Instantiation</td>
<td>1735</td>
</tr>
<tr>
<td></td>
<td></td>
<td>List</td>
<td>627</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Alternative</td>
<td>531</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Expansion</td>
<td>118</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Exception</td>
<td>16</td>
</tr>
<tr>
<td>Comparison</td>
<td></td>
<td>Contrast</td>
<td>5947</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Concession</td>
<td>1425</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Comparison</td>
<td>553</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pragmatic contrast</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pragmatic concession</td>
<td>12</td>
</tr>
<tr>
<td>Contingency</td>
<td></td>
<td>Cause</td>
<td>6203</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Condition</td>
<td>1359</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pragmatic cause</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pragmatic condition</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contingency</td>
<td>2</td>
</tr>
<tr>
<td>Temporal</td>
<td></td>
<td>Asynchronous</td>
<td>2730</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Synchrony</td>
<td>4007</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Temporal</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 4: The PDTB binary, top-level (L1) and fine-grained (L2) discourse relations and their distribution in PDTB 1.0 and 2.0 datasets. L2 relations marked italics were used for 10-class classification with XLNet.

Appendix II: Implicit Claim Reconstruction

<table>
<thead>
<tr>
<th>Topic</th>
<th>Claim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ban of plastic bottles</td>
<td>The consumption of water bottles should not be banned.</td>
</tr>
<tr>
<td></td>
<td>The consumption of water bottles should be allowed only in the case of emergency.</td>
</tr>
<tr>
<td>Christianity or atheism</td>
<td>I choose atheism over Christianity and do not believe in God.</td>
</tr>
<tr>
<td></td>
<td>I choose Christianity over atheism and do believe in God.</td>
</tr>
<tr>
<td>Evolution vs. creation</td>
<td>The world was created by God.</td>
</tr>
<tr>
<td></td>
<td>The evolution is the beginning of life.</td>
</tr>
<tr>
<td>Personal pursuit or advancing the common good?</td>
<td>Advancing the common good is better than personal pursuit.</td>
</tr>
<tr>
<td></td>
<td>Personal pursuit is better than advancing the common good.</td>
</tr>
<tr>
<td>Should physical education be mandatory in schools?</td>
<td>Physical education should not be mandatory in schools.</td>
</tr>
<tr>
<td></td>
<td>Physical education should be mandatory in schools.</td>
</tr>
<tr>
<td>Is TV better than books?</td>
<td>Books are better than TV.</td>
</tr>
<tr>
<td></td>
<td>TV is better than books.</td>
</tr>
</tbody>
</table>

Table 5: Examples of the claims reconstructed based on the corresponding Dagstuhl15512 ArgQuality topics.

Appendix III: Argument Graphs Examples
Figure 6: Examples of the unique discourse-based argument schemes reconstructed from the Dagstuhl15512 ArgQuality corpus and their relative frequencies (in %).
A Hybrid Approach of Opinion Mining and Comparative Linguistic Analysis of Restaurant Reviews

Salim Sazzed
Old Dominion University
Norfolk, VA, USA
ssazz001@odu.edu

Abstract

The existing research on sentiment analysis mainly utilized data curated in limited geographical regions and demography (e.g., USA, UK, China) due to commercial interest and availability of review data. Since the user’s attitudes and preferences can be affected by numerous sociocultural factors and demographic characteristics, it is necessary to have annotated review datasets belong to various demography. In this work, we first construct a review dataset BanglaRestaurant that contains over 2300 customer reviews towards a number of Bangladeshi restaurants. Then, we present a hybrid methodology that yields improvement over the best performing lexicon-based and machine learning (ML) based classifier without using any labeled data. Finally, we investigate how the demography (i.e., geography and nativeness in English) of users affect the linguistic characteristics of the reviews by contrasting two datasets, BanglaRestaurant and Yelp. The comparative results demonstrate the efficacy of the proposed hybrid approach. The data analysis reveals that demography plays an influential role in the linguistic aspects of reviews.

1 Introduction

Sentiment analysis or opinion mining refers to the process of identifying opinions or sentiments expressed (e.g., positive, negative) in a text document (Liu, 2012). The lexicon-based method and machine learning (ML) based method are the two dominant approaches for opinion mining; although, their combinations have been also explored by the researchers. To evaluate the polarity of a piece of text, the lexicon-based methods rely on the sentiment lexicon comprised of opinion-conveying positive or negative terms and a set of rules (Turney, 2002; Sazzed, 2020b). For the lexicon-based methods, the laborious steps of data labeling are not required. The supervised machine learning (ML) based approaches derive the relationship between features of the text segments and the opinions expressed in the writing in a supervised fashion (Pang et al., 2002; Sazzed and Jayarathna, 2019). Therefore, supervised ML classifiers require a significant amount of annotated data for training a predictive model for determining the polarity of a text document. Labeling a large amount of text data is not only a challenging but also a tedious and costly process (Sazzed and Jayarathna, 2021; Sazzed, 2020a).

Mining customer opinions towards restaurants has attained popularity in recent years due to its impact on business growth and sustainability. Researchers performed a number of studies using the Yelp restaurant review and several other datasets. However, existing research on opinion mining in review datasets mainly focused on data curated in some specific demography (e.g., developed countries or countries with large consumer markets) due to commercial interest and abundance of user-generated review data. As shown in (Nakayama and Wan, 2019), the trait and preferences of users can vary across demography. Therefore, it is important to generate annotated content for various demography (e.g., geography, user, or language) and analyze them to identify the differences, which can ultimately help better decision-making. Nowadays, with the increasing accessibility of the Internet and the popularity of social media, opinion data are increasingly becoming available in many other geographies (e.g., Bangladesh) and languages.

Therefore, the main objective of this work is to create a restaurant review dataset from less-explored demography and introduce a new methodology to improve the performance of sentiment
classification. Besides, this study aims to explore how the demography affects the linguistic characteristics of reviews.

We create a restaurant review dataset, BanglaRestaurant, which contains more than 2300 customer reviews toward various Bangladeshi restaurants. We employ both the lexicon-based and ML-based methods to classify customer’s sentiments in the BanglaRestaurant dataset. To improve the performance of sentiment classification, a hybrid methodology is introduced that leverages a lexicon-based method and an ML-based classifier. We observe an improvement of the F1 score by employing the proposed hybrid approach.

We investigate the characteristics of reviews belong to the BanglaRestaurant dataset written by non-native English speakers and Yelp reviews written by English native speakers. The comparative analysis reveals that demography (i.e., nativeness of language and geography) has influences on the various linguistic features of reviews.

1.1 Contributions

The contributions of this paper can be summarized as follows:

- We create a Bangladeshi restaurant dataset consists of over 2300 customer reviews\(^1\).
- We propose a hybrid approach that improves the performance of sentiment classification by combining the lexicon-based method and supervised ML classifier.
- We analyze the characteristics of two restaurant review datasets curated in different demography.

2 Sentiment Analysis in Restaurant Review Datasets

Kang et al. (2012) created a sentiment lexicon and proposed an improved Naive Bayes (NB) based method for sentiment analysis in a restaurant dataset. Blair-Goldensohn et al. (2008) introduced a sentiment summarizer system where a summary is built by extracting relevant aspects of a service, aggregating the sentiment per aspect, and selecting aspect-relevant text. An attention-based Long Short-Term Memory (LSTM) network was proposed in (Wang et al., 2016) for aspect-level

\(^1\)https://github.com/sazzadcsedu/BanglaRestaurant.git

sentiment classification. Gan et al. (2017) analyzed how various attributes influence customers sentiments on restaurant star ratings. Zhang et al. (2011) incorporated ML-based techniques such as NB and SVM to automatically classify user reviews as positive or negative from online Cantonese-written restaurant reviews.

Zahoor et al. (2020) created a restaurant dataset by collecting over 4000 customer reviews of various restaurants located in Pakistan. Sasmita et al. (2017) performed aspect-based sentiment analysis (ABSA) in Indonesian restaurant reviews. They performed both the (i) aspect extraction and (ii) aspect sentiment orientation classification.

Xue et al. (2017) identified aspect categories and extracted aspect-terms from the user-generated reviews. The authors proposed a multi-task learning model based on neural networks and observed improved performance over the models trained separately on three public datasets. Ahiladas et al. (2015) utilized named entity recognition (NER) and typed dependency techniques to identify different types of food and the opinions associated with them. Tian et al. (2021) performed a case study on Yelp restaurant review data to find what affects restaurant customer’s sentiments. Besides, they noticed consumers rate restaurant service more often than the food quality. Jia (2018) proposed an integrated approach that leverages text mining and empirical modeling to correlate ratings with reviews. The author examined 49,080 pairs of restaurant ratings and reviews from Dianping.com (a Chinese online review community) to identify high-frequency words, major topics, and subtopics. Xiang et al. (2019) presented an LSTM based architecture LSTM-SAT for sentiment analysis of Cantonese style text by incorporating sentiment knowledge into the attention mechanism in the LSTM.

3 BanglaRestaurant Dataset

3.1 Data Collection

The restaurant reviews are manually collected from the restaurant’s Facebook pages. We find the reviews are written in English, Bengali (i.e., the native language of Bangladesh), and Romanized-Bengali (Bengali text in Latin alphabet). English is the main foreign language in Bangladesh which is taught in schools and colleges. Besides, English is frequently used in government administration, educational institutions, courts, businesses, and media of the country. People often use English for
expressing their opinions and feelings on social media as it is more convenient to write English text than Bengali. For example, Bengali has 50 letters (11 vowels and 39 consonants) compared to 26 letters in English. Since this study focuses on the reviews written in English, the final dataset contains only the English reviews.

### 3.2 Data Annotation

We annotate the reviews based on the reviewer’s recommendations; if the reviewer recommends the restaurant, then the corresponding review belongs to the positive class; Otherwise, it goes to the negative class. The final restaurant dataset contains a total of 2315 reviews, 1702 positive reviews (i.e., recommended by customer), and 613 negative reviews (not recommended).

### 4 Proposed Methodology

To determine the semantic orientations of the reviews, we employ both the lexicon-based and ML-based methods, as each of them has certain advantages over the other.

#### 4.1 Lexicon-Based Approaches

We employ four lexicon-based methods: VADER (Hutto and Gilbert, 2014), TextBlob, LRSentiA (Sazzed and Jayarathna, 2021) and SentiStrength (Thelwall et al., 2010) for classifying sentiment from unlabeled data. A non-negative polarity score by these methods is considered as a positive prediction (except VADER, where the compound score is used).

##### 4.1.1 VADER

VADER (Valence Aware Dictionary for Sentiment Reasoning) is a lexicon and rule-based sentiment analysis tool specifically attuned to determine sentiments expressed in social media. In VADER, a compound score indicates the semantic orientation of a review. For binary classification, a non-negative compound score refers to a positive prediction.

##### 4.1.2 LRSentiA

LRSentiA is a lexicon and rule-based method that can classify opinions expressed in unlabeled data. LRSentiA utilizes a binary-level opinion lexicon (Liu, 2010) and a set of linguistic rules to determine the polarity of a review.

##### 4.1.3 TextBlob

TextBlob is a Python library for processing textual data. The predicted polarity score of a review is within the range of \([-1, +1]\), where -1 indicates strongly negative, and +1 means strongly positive.

##### 4.1.4 SentiStrength

SentiStrength predicts the strength of positive and negative sentiments in short texts. The range of sentiment value of negative sentiment could be between -1 to -5; For the positive sentiment, the score can range between +1 and +5.

#### 4.2 Machine Learning Approaches

##### 4.2.1 ML Classifiers

In this work, we employ five popular supervised ML classifiers: Logistic Regression (LR), Ridge Regression (RR), Support Vector Machine (SVM), Random Forest (RF), and Extra Tree Classifier (ET) for identifying the polarity of the customer review.

##### 4.2.2 Experimental Settings

The review texts are segmented into words and converted to a matrix of term frequency-inverse document frequency (TF-IDF) features. We calculate the TF-IDF score for all words in the review using the scikit-learn library (Pedregosa et al., 2011), and the resultant matrix is feed to supervised ML classifiers. To evaluate the performance of various ML classifiers, we use 10-fold cross-validation. For all the ML classifiers, the default parameter settings of the scikit-learn library (Pedregosa et al., 2011) with class-balanced weights are used.

#### 4.3 Hybrid Approach

We present a hybrid methodology for sentiment classification by leveraging both the lexicon-based and ML-based approaches. Based on the predicted polarity scores of the reviews determined by the lexicon-based method LRSentiA (Sazzed and Jayarathna, 2021), we categorize them into three groups.

1. **Minimal opinion group (MOG)**: When LRSentiA assigns a polarity score of 0 to a review, it falls into the MOG category. A review with a 0 polarity score is considered as a positive prediction assuming that a negative review has a higher chance of having negative polarity scores than a positive review with a positive polarity score. Thus, predictions with non-negative polarity scores are considered positive predictions.
2. *Fair opinion group* (FOG): When the predicted polarity score of a review is between $<-2,+2>$, it belongs to the FOG group.

3. *Strong opinion group* (SOG): The reviews with polarity scores above +2 or less than -2 fall into SOG.

We assume if a lexicon-based method predicts the class of a review with a high polarity score (i.e., highly positive (> 2) or highly negative score (< -2)), it is highly probable that prediction is correct. As the lexicon-based method relies on the polarity of individual opinion words, if the overall polarity score is strongly positive or negative, then the review consists of mostly positive aspects (high positive score) or negative aspects (high negative score); thus, the prediction is probably correct.

After excluding the reviews belong to *strong opinion group* (SOG), the remaining two groups contain reviews which the lexicon-based method cannot distinguish confidently based on the opinion words present in the review. This scenario could happen due to various reasons, such as lexicon coverage problems or the complexity of natural languages. These groups are more prone to misclassification by the lexicon-based methods.

ML classifiers have been successfully applied in numerous problems in varying domains when data is noisy or explicit rules can not separate the classes well. Since supervised ML classifiers are capable of characterizing the best mapping from input to output, we employ an ML classifier for the reviews that require learning the implicit pattern from the labeled data rather than just using the polarity of the individual opinion words to find overall sentiment.

The overall predictions by the hybrid method consist of predictions by LRSentiA for the highly polar reviews and an ML classifier for the remaining reviews.

5 Impact of Demography in Linguistic Attributes of Review

We analyze whether the demography of the reviews has any impact on the linguistic characteristics of the reviews. We consider two datasets, which differ in the following demographic aspects, geography (i.e., the location where reviews were written) and language nativeness of speakers (i.e., whether native or non-native speakers of English wrote reviews). Our developed *BanglaRestaurant* corpus is curated in Bangladesh, where people are non-native speakers of English (sample reviews shown in Figure 2). We contrast this dataset with the *Yelp* restaurant reviews, which were written by USA-based (mostly) English native speakers (sample reviews shown in Figure 3).

The *Yelp* dataset contains 6860 positive comments and 1676 negative comments in contrast to 1702 positive samples and 613 negative samples in *BanglaRestaurant* dataset. To avoid any kind of influence of class distribution and dataset size, we use the same number of reviews from both the *BanglaRestaurant* and *Yelp* datasets. For the *Yelp* dataset, we randomly shuffle and then select 1702 positive samples and 613 negative reviews out of 6860 positive and 1676 negative reviews present.
We analyze the following characteristics of the reviews in two datasets. For the Yelp dataset, the numbers represent the average results of five random selections of 1702 positive comments and 613 negative comments.

1. Word count of the corpus: The numbers of words present in both corpora are provided.
2. Sentence count in the corpus: We report the number of sentences present in both datasets.
3. Average review length (word-level): The average review length in word-level indicates the average number of words present in a review of a corpus.
4. Average review length (sentence-level): The average review length in sentence-level refers to the average number of words present in a review of a corpus.
5. Average sentence length: This metric provides the average number of words each sentence contains in reviews that belong to a corpus.
6. Coverage of a lexicon: Furthermore, we compute the lexicon coverage of two English lexicons (Liu, 2010) and (Hutto and Gilbert, 2014) in both datasets. The lexicon coverage can assess the presence of diverse opinion words in the reviews, which indicates the vocabulary range of users.
7. Usage of the complex sentence in reviews: Besides, we study the complexity of the reviews at the sentence level. A complex sentence usually contains one or more dependent (subordinate) clauses and one or more independent clauses. A subordinating conjunction is a word or phrase that connects a dependent clause to an independent clause. Some examples of subordinating conjunctions are, although, as, because, before, how, if, once, since, etc.. We examine the presence of 50 common subordinating conjunctions in both corpora to analyze the complexity of the reviews.

### 6 Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Pre.</th>
<th>Rec.</th>
<th>F1</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SentiStrength</td>
<td>0.896</td>
<td>0.789</td>
<td>0.839</td>
<td>88.0%</td>
</tr>
<tr>
<td>TextBlob</td>
<td>0.896</td>
<td>0.821</td>
<td>0.857</td>
<td>88.7%</td>
</tr>
<tr>
<td>VADER</td>
<td>0.895</td>
<td>0.824</td>
<td>0.858</td>
<td>89.2%</td>
</tr>
<tr>
<td>LRSentiA</td>
<td>0.901</td>
<td>0.822</td>
<td>0.860</td>
<td>89.5%</td>
</tr>
<tr>
<td>ET</td>
<td>0.846</td>
<td>0.832</td>
<td>0.834</td>
<td>87.9%</td>
</tr>
<tr>
<td>RF</td>
<td>0.855</td>
<td>0.833</td>
<td>0.840</td>
<td>88.0%</td>
</tr>
<tr>
<td>LR</td>
<td>0.878</td>
<td>0.904</td>
<td>0.891</td>
<td>91.4%</td>
</tr>
<tr>
<td>RR</td>
<td>0.884</td>
<td>0.901</td>
<td>0.893</td>
<td>91.7%</td>
</tr>
<tr>
<td>SVM</td>
<td>0.882</td>
<td>0.903</td>
<td>0.893</td>
<td>91.5%</td>
</tr>
</tbody>
</table>

Table 1: Performances of Various Lexicon-based and ML-based Methods in BanglaRestaurant Dataset

<table>
<thead>
<tr>
<th>Group</th>
<th>Pre.</th>
<th>Rec.</th>
<th>F1</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOG</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>61.65%</td>
</tr>
<tr>
<td>FOG</td>
<td>0.86</td>
<td>0.84</td>
<td>0.85</td>
<td>87.04%</td>
</tr>
<tr>
<td>SOG</td>
<td>0.97</td>
<td>0.95</td>
<td>0.96</td>
<td>97.58%</td>
</tr>
<tr>
<td>Overall</td>
<td>0.90</td>
<td>0.82</td>
<td>0.86</td>
<td>89.5%</td>
</tr>
</tbody>
</table>

Table 2: Performance of The Best Lexicon-based Method LRSentiA in BanglaRestaurant Dataset

Table 1 reveals that TextBlob, VADER, and LRSentiA perform similarly, where the SentiStrength yields comparatively lower F1 score and accuracy. SVM, LR, and RR provide a similar F1 score of around 0.89 and an accuracy of 91%, which is a bit higher than the top lexicon-based method, LRSentiA. Decision tree-based ML classifiers such as DT and ET provide comparatively low accuracy and F1 score compared to other ML classifiers.

3https://github.com/sazzadcsedu/50SubordinateConjunctions.git
Table 3: The Various Linguistics Attributes of Reviews Belong to Yelp and BanglaRestaurant Datasets

<table>
<thead>
<tr>
<th>Features</th>
<th>BanglaRestaurant</th>
<th>Yelp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of words in corpus</td>
<td>61523</td>
<td>295781.2</td>
</tr>
<tr>
<td>Total number of sentences in corpus</td>
<td>7258</td>
<td>22588.6</td>
</tr>
<tr>
<td>Avg. number of words/review</td>
<td>26.575</td>
<td>127.767</td>
</tr>
<tr>
<td>Avg. number of sentences/review</td>
<td>3.1352</td>
<td>9.757</td>
</tr>
<tr>
<td>Avg. number of words/sentence</td>
<td>8.47</td>
<td>13.09</td>
</tr>
<tr>
<td>Number of opinion words (Hu-Liu)</td>
<td>4377</td>
<td>16112.4</td>
</tr>
<tr>
<td>Lexicon coverage (Hu-Liu)</td>
<td>7.1%</td>
<td>5.4%</td>
</tr>
<tr>
<td>Number of opinion words (VADER)</td>
<td>4655</td>
<td>16429.6</td>
</tr>
<tr>
<td>Lexicon coverage (VADER)</td>
<td>7.56%</td>
<td>5.55%</td>
</tr>
<tr>
<td>Subordinating conjunctions in corpus</td>
<td>1214(61523)</td>
<td>8704.4(295781.2)</td>
</tr>
<tr>
<td>Subordinating conjunctions per reviews</td>
<td>0.52</td>
<td>3.76</td>
</tr>
</tbody>
</table>

Table 4: Accuracy and F1 Scores of Various ML Classifiers in MOG and FOG Groups (990 Reviews)

<table>
<thead>
<tr>
<th>Method</th>
<th>F1 Score</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRSentiA</td>
<td>0.755</td>
<td>79.6%</td>
</tr>
<tr>
<td>ET</td>
<td>0.744</td>
<td>79.6%</td>
</tr>
<tr>
<td>RF</td>
<td>0.776</td>
<td>78.8%</td>
</tr>
<tr>
<td>RR</td>
<td>0.831</td>
<td>85.1%</td>
</tr>
<tr>
<td>LR</td>
<td>0.840</td>
<td>85.7%</td>
</tr>
<tr>
<td>SVM</td>
<td>0.845</td>
<td>86.3%</td>
</tr>
</tbody>
</table>

Table 5: Accuracy and F1 Scores of The Hybrid Approach Integrating Two Best Performing ML Classifiers

<table>
<thead>
<tr>
<th>Method</th>
<th>F1 Score</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid-LR</td>
<td>0.902</td>
<td>92.5%</td>
</tr>
<tr>
<td>Hybrid-SVM</td>
<td>0.915</td>
<td>92.75%</td>
</tr>
</tbody>
</table>

Table 2 exhibits that lexicon-based method LRSentiA fails to classify the reviews correctly in many cases when the predicted polarity score is low (i.e., between -2 and +2, inclusive). In the MOG and FOG groups, only 779 out of 990 reviews are classified correctly. We notice when the lexicon-based method predicts with high polarity score, it is accurate in most cases. Among 1325 reviews, the predictions of LRSentiA are true for 1293 cases with an accuracy of 97.5%.

Table 4 presents the performance of ML classifiers in the complex subsets (MOG and FOG) of BanglaRestaurant reviews (i.e., 990 reviews out of 2315), which the lexicon-based method fails to not discern correctly. We find that most of the ML classifiers yield better performance compared to the lexicon-based classifier. The best performing classifier SVM increases F1 score 12% over the lexicon-based method in the MOG and FOG confidence groups.

Table 5 shows that the proposed hybrid method enhances the F1 score and accuracy of classification by integrating an ML classifier such as LR or SVM with the lexicon-based method. While the best lexicon-based and ML-based methods show F1 scores of 0.86 and 0.893, respectively, the hybrid approach incorporating the SVM classifier attains an F1 score of 0.91.

From Table 3, we observe that Yelp review lengths are much higher in both word and sentence levels. The sentiment lexicon shows higher coverage in BanglaRestaurant; The presence of the subordinate clause, which refers to the complex sentence, is more obvious in the Yelp dataset.

7 Discussion

We observe in the BanglaRestaurant dataset performance of the lexicon-based approach is close to the ML-based classifiers. The best macro F1 score is obtained from the SVM classifier, which is around 89%; The most accurate lexicon-based method LRSentiA achieves an accuracy of 86%. This result is expected as supervised ML classifiers usually perform better than lexicon-based methods.

From Table 2, it is evident that the lexicon-based method is very effective when the review polarity is easily distinguishable either as positive or negative. If a user review is comprised of mixed opinions towards various entities or sentiment is not obvious, it is often difficult to assign the overall polarity using the lexicon-based method. In contrast, the ML classifiers learn implicit patterns from training data, thus, are capable of determining the overall sentiment of a review even though the opinion is
not apparent. Thus combining both the lexicon and ML-based classifiers in the sentiment classification framework improves the performance.

We notice the hybrid approach yields an overall F1 score of 0.915 in the BanglaRestaurant corpus, an improvement of 6% (0.860) over the best lexicon-based method and 3% (0.892) over the best ML-based classifier. Although the increase of the F1 score is not much compared to the best ML classifier, the main advantage of the proposed hybrid approach is that it does not require any annotated data. Review data is usually readily available on the web; the primary challenge is to label the data. Thus, the proposed hybrid approach can be very effective for addressing the data annotation difficulties. We find the features characteristics of the reviews are distinct in BanglaRestaurant and Yelp datasets, which represent data from different demography.

8 Summary and Conclusions

In this work, we introduce a hybrid approach for sentiment classification in a newly created BanglaRestaurant dataset. The proposed hybrid approach combines the lexicon-based method LRSentiA with the SVM classifier to improve the performance of sentiment classification. The results suggest that lexicon-based methods are mainly effective at classifying reviews that contain strong opinions. However, they struggle to determine sentiment when the polarity is not obvious. Hence, it is necessary to incorporate an ML classifier that is robust for complex cases.

In addition, we provide a comparative analysis of review data curated in different demography. We investigate various linguistic features of reviews that belong to these two datasets. The first dataset contains (i.e., BanglaRestaurant) reviews written by non-native English speakers of Bangladesh; while, reviews of the other dataset (i.e., Yelp) were written by (mostly) English native speakers located in the USA. We observe differences in the various linguistic characteristics of reviews in these two datasets.

Acknowledgments

The author likes to thank Md. Samiul Basir Tasin and MD Shafin Islam Rudro for collecting BanglaRestaurant review data. The conference registration fee was supported by the ISAB VISA Scholarship of Old Dominion University.

References


Sasha Blair-Goldensohn, Kerry Hannan, Ryan T. McDonald, Tyler Neylon, George A. Reis, and Jeff Reynar. 2008. Building a sentiment summarizer for local service reviews.


A Lexicon for Profane and Obscene Text Identification in Bengali

Salim Sazzed
Old Dominion University
Norfolk, VA, USA
ssazz001@odu.edu

Abstract

Bengali is a low-resource language that lacks tools and resources for profane and obscene textual content detection. Until now, no lexicon exists for detecting obscenity in Bengali social media text. This study introduces a Bengali obscene lexicon consisting of over 200 Bengali terms that can be considered filthy, slang, profane or obscene. A semi-automatic methodology is presented for developing the obscene lexicon that leverages an obscene corpus, word embedding, and part-of-speech (POS) taggers. The developed lexicon achieves coverage of around 0.85 for obscene and profane content detection in an evaluation dataset. The experimental results imply that the developed lexicon is effective at identifying obscenity in Bengali social media content.

1 Introduction

The popularity of e-commerce and social media has surged the availability of user-generated content online. Therefore, text analysis tasks such as sentiment classification (Feldman, 2013; Sazzed and Jayarathna, 2019; Yadollahi et al., 2017; Sazzed, 2021b; Sazzed and Jayarathna, 2021), hate speech detection (Poletto et al., 2021; Corazza et al., 2020), profane or abusive content identification (Caselli et al., 2020; Nobata et al., 2016) have received significant attention in recent years. Profanity indicates the usage of taboo or swearing words and is prevalent in social media data across languages (Wang et al., 2014). The presence of swearing, obscene or vulgar words could be linked with hate speech, sexism, and racism. Hence, identifying their presence is important to understanding and monitoring online content. Although the terms profanity, obscenity, swearing, and vulgarity have subtle differences in their meaning, they are closely connected with some overlapping definitions. Thus, in this paper, they have been used interchangeably to refer to filthy content.

A lexicon consisting of a list of words with specific annotations can play an important role in various natural language processing tasks, such as sentiment analysis or inappropriate content identification. A profane or obscene lexicon contains words that convey foul, filthy, and profane meanings (e.g., ass, bitch). An obscene lexicon is instrumental for determining profanity, vulgarity, or obscenity in a text. The presence of swearing in English social media has been investigated by various researchers (Wang et al., 2014; Pamungkas et al., 2020). Wang et al. (2014) found that the rate of swear word use in English Twitter is 1.15%, almost double compared to its use in daily conversation (0.5% - 0.7%) as observed in previous work (Jay, 1992). Wang et al. (2014) also reported that in a random sampling, they observed around 7.73% tweets containing swear words. Furthermore, vulgar word identification can help to improve sentiment classification as shown by various researchers (Volkova et al., 2013; Yang and Eisenstein, 2017).

In Bengali, although few works performed abusive content analysis, none of them focused on determining obscenity or generating resources for identifying obscenity. Until now, no lexicon exists in Bengali that can help to identify profanity in text data. Thus, in this work, the goal is to generate resources for obscenity identification.

To construct the Bengali obscene lexicon, we propose a corpus-based semi-automatic approach. From an existing Bengali obscene corpus, utilizing word embedding and POS tagging, the lexicon is created. To demonstrate the efficacy of this lexicon, we categorize a drama review corpus into profane and non-profane categories based on the presence of swear and obscene terms. We observe that the developed lexicon successfully identifies 85.5% of the obscene or profane reviews in the corpus.
1.1 Motivation and Contributions

With the rapid growth of user-generated Bengali content on social media and the web, the presence of inappropriate content has become an issue. The content which is not in line with the social norms and expectations of a community needs to be censored. In Bengali, no such resources exist; thus, we focus on building a lexicon consisting of swear or obscene words that can help to identify profane content.

The main contributions of this paper can be summarized as follows:

- We introduce a Bengali obscene lexicon comprised of about 200 swear words. We have made the developed lexicon publicly available for researchers.

- We present a semi-automatic methodology for developing a swear lexicon utilizing an obscene corpus and various natural language processing tools.

- We demonstrate that the developed lexicon is effective at profanity detection in Bengali social media content.

2 Related Work

The existence and socio-linguistics characteristics of swearing or cursing in social media have been studied in several studies. Wang et al. (2014) investigated the ubiquity, utility, and contextual dependency of swearing on Twitter. Gauthier et al. (2015) analyzed several sociolinguistic aspects of swearing on Twitter text data. Several studies investigated the relationship between social factors, such as gender with the profanity, and discovered males employ profanity much more often than females (Wang et al., 2014; Selnow, 1985). Other social factors such as age, religiosity, or social status were found to be related to the rate of using vulgar words (McEnery, 2004). Jay and Janschewitz (2008) noticed that the offensiveness of taboo words depends on their context, and found that usages of taboo words in conversational context is less offensive than hostile context. Pinker (2007) classified the use of swear words into five categories: dysphemistic; abusive, using taboo words to abuse or insult someone; idiomatic, using taboo words to arouse the interest of listeners without really referring to the matter; emphatic, to emphasize another word; cathartic, the use of swear words as a response to stress or pain.

Obscenity and profanity filtering has been studied for content filtering, such as parental controls (Weir and Duta, 2012), cyberbullying detectors (Dadvar et al., 2013). A more complex application of obscenity filtering is identifying implicitly abusive content, where both the intention of the author and the usage of obscene language need to be considered (Weir and Duta, 2012).

Research related to the identification of swearing or offensive words has been conducted mainly in English; Therefore, lexicons comprised of offensive words are available in the English language. Pamungkas et al. (2020) created SWAD (Swear Words Abusiveness Dataset), a Twitter English corpus, where abusive swearing is manually annotated at the word level. Their collection consists of 1,511 unique swear words from 1,320 tweets. Razavi et al. (2010) manually collected approximately 2,700 dictionary entries, including phrases and multi-word expressions, which is one of the earliest work offensive lexicon creations. The recent work of lexicon creation for hate speech detection was reported in (Gitari et al., 2015). Another English lexicon of abusive words was provided by (Wiegand et al., 2018).

Eder et al. (2019) explored the vulgar and obscene text in German. They conceived vulgar language is predominantly signaled by an overly lowered language, disgusting and obscene lexicalizations, which is generally banned from any type of civilized discourse. Primarily, it refers to sexual organs and activities, as well as body parts and scatological expressions.

In Bengali, several works investigated the presence of abusive language in social media data by employing supervised ML classifiers and labeled data (Ishmam and Sharmin, 2019; Banik and Rahman, 2019). Emon et al. (2019) utilized linear support vector classifier (LinearSVC), logistic regression (LR), multinomial naive Bayes (MNB), random forest (RF), artificial neural network (ANN), recurrent neural network (RNN) with long short term memory (LSTM) to detect multi-type abusive Bengali text. They found RNN outperformed other classifiers by obtaining the highest accuracy of 82.20%.

Chakraborty and Seddiqui (2019) employed ma-
chine learning and natural language processing techniques to build an automatic system for detecting abusive comments in Bengali. As input, they used Unicode emoticons and Unicode Bengali characters. They applied MNB, SVM, Convolutional Neural Network (CNN) with LSTM and found SVM performed best with 78% accuracy. Sazzed (2020b) created a sentiment lexicon that consists of over 500 Bengali negative opinion words. However, no annotations regarding obscenity or vulgarity were provided for these negative words.

Karim et al. (2020) proposed BengFastText, a word embedding model for Bengali, and incorporated it into a Multichannel Convolutional-LSTM (MConv-LSTM) network for predicting different types of hate speech. They compared BengFastText against the Word2Vec and GloVe embedding by integrating them into several ML classifiers and showed the efficacy of BengFastText for hate speech detection.

Sazzed (2021a) introduced an annotated Bengali corpus of 3000 transliterated Bengali comments categorized into two classes, abusive and non-abusive, 1500 comments for each. For the baseline evaluations, the author employed several supervised machine learning (ML) and deep learning-based classifiers. They observed support vector machine (SVM) shows the highest efficacy for identifying abusive content.

To the best of the author’s knowledge, none of these existing works concentrated on recognizing obscene words in Bengali social media content. Besides, no lexicon exists so that profanity or obscenity can be determined without using any annotated data. This work is the first effort to identify profanity in the context of Bengali social media data by introducing a obscene lexicon.

3 Corpora

Two Bengali obscene datasets are used in this study, one for constructing the lexicon (development corpus), and the other one is for evaluating the performance of the created lexicon (evaluation corpus).

3.1 Development Corpus

The development corpus is a subset of a Bengali corpus deposited by (Abu, 2020). This Bengali corpus consists of 10221 user comments which belong to different categories, such as toxic, racism, obscene, insult, etc.

For developing the lexicon, only the obscene comments are used. After discarding the noisy reviews (e.g., empty comments, punctuation only comments, etc.) and reviews that belong to other classes, the development corpus consists of 3902 obscene comments (each contains 1-100 words).

Figure 1 represents some examples of the obscene comments from the development corpus.

3.2 Evaluation Corpus

The evaluation corpus is a subset of the dataset deposited by Sazzed (2020a). This corpus consists of viewer’s comments towards a number of Bengali dramas collected from Youtube. Originally, this dataset contains 8500 positive and 3307 negative reviews. These 3307 negative reviews are further categorized into two classes, obscene and non-obscene. After annotation, this corpus consists of 2643 non-obscene reviews and 664 obscene reviews.

4 Creation of Obscene Lexicon

4.1 Text Processing Tools

4.1.1 POS Tagger

Part-Of-Speech (POS) tagger is a text analysis tool that assigns a POS tag (e.g., noun, verb, adjective, etc.) to each word of a piece of text data. As adjectives, nouns, and verbs usually convey opinions, the POS tagger can help to identify words that convey obscenity. Some of the popular POS taggers in English are NLTK POS tagger (Loper and Bird, 2002), spaCy POS tagger (Honnibal and Montani, 2017), etc.

We utilize a Bengali POS tagger to identify opinion conveying word (i.e., adjective and verb). However, the existing Bengali POS taggers are not as accurate as of its English counterpart. Hence, manual validation is needed to check the correctness of the POS tags assigned to words.

4.1.2 Word Embedding

The word embedding is a learned representation for textual content. A word embedding creates similar representations of words that are related in some ways. The word-embedding provides an efficient way to use the dense representation of words of varying lengths.
There exist two main approaches for learning word embedding, count-based and context-based. The count-based vector space models heavily rely on the word frequency and co-occurrence matrix with the assumption that words in the same contexts share similar or related semantic meanings. The other learning approach, context-based methods, build predictive models that predict the target word given its neighbors. The best vector representation of each word is learned during the model training process.

The continuous Bag-of-Words (CBOW) model is a popular context-based method for learning word vectors. It predicts the center word from surrounding context words.

4.2 Lexicon Creation Framework

Leveraging lexical resources can assist in identifying the presence of profanity in Bengali social media. This study presents a semi-automatic approach for creating a swear lexicon utilizing an annotated corpus, word-embedding, and POS tagger. The lexicon development framework consisting of three phases, as shown in Figure 2,

1. Selection of seed words
2. Expansion of lexicon
3. Manual validation

4.2.1 Selection of Seed Words

The proposed methodology adopts a labeled obscene corpus to generate a list of seed words. The occurrences of individual words in the corpus are counted. Based on the word-occurrence count, the top 100 words are selected. We observe the presence of some non-vulgar words among the top 100 words, which are excluded.

4.2.2 Expansion of Lexicon

The lexicon expansion step involves utilizing word embedding to identify similar words of the seed list. The Gensim (Řehůřek and Sojka, 2010) continuous Bag-of-Words (CBOW) implementation is used to find similar words in the development corpus.

The entire procedure consists of the following steps.

- In the first step, we identify words that are the most similar to the seed words.
- The second step iteratively finds words similar to obscene words recognized in the first step. The duplicate words are removed automatically. In addition, we exclude words that are not a noun, adjective, or verb. This iteration stops when we notice no significant expansion of the obscene word list.

4.2.3 Manual Validation

In the final step, we manually exclude non-obscene words that exist in the lexicon. As lexical resources
Figure 3: Examples of obscene/swear words from the created lexicon, such as POS tagger in Bengali are not sophisticated enough, a manual validation step is necessary to eliminate non-obscene words. Moreover, we find that vulgar comments often do not follow the usual sentence structure; therefore, the POS tagger often fails to tag them correctly.

4.3 Developed Lexicon

Figure 3 shows some examples of Bengali obscene words and corresponding English translations. The English translation is provided to give an idea of the type/characteristics of the lexicon. Due to language-specific differences, the exact English translation may not be available for some of the words present in the developed obscene lexicon.

5 Obscenity Detection in Textual Content

As no lexicon exists in Bengali for obscene content detection, we compare the performance of the developed lexicon with several supervised classifiers in the evaluation corpus.

5.1 ML Classifiers

Two supervised ML classifiers, Logistic Regression (LR) and Support Vector Machine (SVM), and an optimizer, Stochastic Gradient Descendent (SGD), are employed in the evaluation corpus to identify profane reviews.

Logistic regression (LR) is a predictive analysis model that assigns observations into a discrete set of classes. LR assumes there are one or more independent variables that determine the outcome of the target.

Support Vector Machine (SVM) is a discriminative classifier defined by a separating hyperplane. Given the labeled training data, SVM generates an optimal hyperplane that categorizes unseen observations. For example, in two-dimensional space, this hyperplane is a line dividing a plane into two parts where each class lays on either side.

SGD is an optimization technique and does not correspond to a specific family of machine learning models. It is only a way to train a model. For SGD, we use hinge loss and l2 penalty.

As a feature vector, we use the unigram and bigram-based tf-idf score. 10-fold cross-validation is performed to assess the performance of various ML classifiers. We use the scikit-learn (Pedregosa et al., 2011) library implementation of the above-mentioned classifiers. For all the classifiers, default parameter settings are used.

To evaluate the performance of various ML classifiers and the SGD optimizer, we employ them in both original class-imbalanced and modified class-balanced settings. For class-balancing, initially, we employ the sampling method, SMOTE (Synthetic Minority Oversampling Technique) (Chawla et al., 2002). SMOTE is an oversampling technique where the synthetic samples are generated for the minority class with the help of interpolation. We notice it can not eliminate the inequality of class distribution in the original dataset. Thus, we use the subsampling method to make the dataset class-balanced.

In the original class-imbalanced setting, all the 664 obscene comments and 2643 non-obscene negative comments are utilized. In the modified class-balanced setting, all the 664 obscene comments are used; however, for the non-obscene class, we randomly select 664 non-obscene comments from a set of 2643 non-obscene comments.

5.2 Evaluation Metric

To show the effectiveness of the created lexicon, we utilize document-level coverage, $DCov$. $DCov$ of a lexicon corresponding to a review corpus is calculated as follows:

First, the number of reviews containing at least one word from the created lexicon is counted; afterward, it is divided by the total number of reviews present in the corpus. The following equation is used to calculate $DCov$ of a lexicon corresponding to a corpus:
Table 1: Comparative Performance of Various Methods for Obscene Text Identification

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Correctly Identified (Out of 664)</th>
<th>DCov/Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexicon-based</td>
<td>Developed Lexicon</td>
<td>564</td>
<td>0.8493</td>
</tr>
<tr>
<td>ML Classifier (Unbalanced)</td>
<td>LR</td>
<td>400</td>
<td>0.6024</td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>399</td>
<td>0.6009</td>
</tr>
<tr>
<td></td>
<td>SGD</td>
<td>384</td>
<td>0.5783</td>
</tr>
<tr>
<td>ML Classifier (Balanced)</td>
<td>LR</td>
<td>609</td>
<td>0.9171</td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>594</td>
<td>0.8945</td>
</tr>
<tr>
<td></td>
<td>SGD</td>
<td>589</td>
<td>0.8870</td>
</tr>
</tbody>
</table>

The main motivation behind creating the obscene lexicon is to identify comments and reviews that contain swearing, profane, or obscene words; thus, DCov is shown only for recognizing the usage of dirty language. Besides, the developed lexicon is manually validated at the final step; thus, it primarily contains obscene or profane words; hence, there is a very low chance that it identifies a non-obscene comment as obscene (false positive). However, this scenario could occur for few words present in the lexicon when they are used in different contexts. For example, one of the Bengali words in the lexicon may refer to either tits or milk, depending on the context.

For the ML classifiers, DCov depicts the recall score for obscene class detection.

5.3 Comparison Results

Table 1 shows that among the 664 obscene reviews present in the evaluation corpus, the developed lexicon registers 564 reviews as obscene by recognizing the presence of at least one obscene/swear/profane term in the review, which is a document-level coverage of around 0.85.

Table 1 provides the coverage of various ML classifiers in the evaluation corpus. We present their performances in two different settings: original class-imbalanced setting and modified class-balanced setting.

From the Table 1, we observe that when the original class-imbalanced data is used, all the three ML classifiers achieve coverage of only 0.60. However, when a class-balanced dataset is utilized, the coverages of classifiers dramatically increase; they achieve around 0.90 coverage.

6 Discussion

The results reveal that the developed lexicon is capable of identifying obscene content in Bengali social media. It shows higher document-level coverage than in-domain labeled data in original class-imbalanced settings (i.e., when the dataset contains mostly non-obscene comments). However, in a class-balanced dataset, we find ML classifiers and the SGD optimizer perform better than the developed lexicon.

Labeled data are scarcely available in low-resource languages such as Bengali; therefore, the developed lexicon can be a practical resource for obscenity identification when labeled data are unavailable. Besides, as shown in Table 1, the performances of ML classifiers can be affected by the class distribution of the training dataset. The obscene or vulgar comments usually occupy only a small portion of a dataset. The ML classifiers can be less effective with the presence of a small sample size of obscene comments. The developed lexicon can be very effective in this scenario.

7 Summary and Conclusion

This study presents a semi-automatic methodology for creating a lexical resource (i.e., an obscene lexicon) to detect obscene content in Bengali. An obscene corpus and various text processing tools and resources are leveraged to develop the obscene lexicon. The developed lexicon is made publicly available. The proposed methodologies can be adapted to other resource-limited languages to create lexical resources.

The efficacy of the obscene lexicon suggests that it can be utilized to distinguish obscenity in Bengali social media content when annotated data are

\[ DCov = \frac{\#\text{reviews with } (>0) \text{ obscene word identified}}{\text{total number of reviews in corpus}} \]

https://github.com/sazzadcsedu/Bangla-Vulgar-Lexicon.git
unavailable. However, it should be noted that labeling textual content as obscene or vulgar entirely based on obscene or swear words may not be sufficient due to the complexity of the natural languages. Still, a well-annotated lexicon of moderate size can assist in identifying profane content; especially, in resource-scarce language. Our future work will involve expanding the size of the lexicon by utilizing larger and multi-domain development corpora.

Acknowledgments

The conference registration fee was supported by the ISAB VISA Scholarship of Old Dominion University.

References


Puja Chakraborty and Md Hanif Seddiqui. 2019. Threat and abusive language detection on social media in bengali language. In 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), pages 1–6. IEEE.


A Case Study of Deep Learning-Based Multi-Modal Methods for Labeling the Presence of Questionable Content in Movie Trailers

Mahsa Shafaei*, Christos Smailis*, Ioannis A. Kakadiaris, Thamar Solorio
Department of Computer Science, University of Houston, Houston, TX, USA
mshafaei@uh.edu, csmailis@central.uh.edu, ikakadia@central.uh.edu, tsolorio@uh.edu
* These authors contributed equally to this work.

Abstract

In this work, we explore different approaches to combine modalities for the problem of automated age-suitability rating of movie trailers. First, we introduce a new dataset containing videos of movie trailers in English downloaded from IMDB and YouTube, along with their corresponding age-suitability rating labels. Secondly, we propose a multi-modal deep learning pipeline addressing the movie trailer age suitability rating problem. This is the first attempt to combine video, audio, and speech information for this problem, and our experimental results show that multi-modal approaches significantly outperform the best mono and bimodal models in this task.

1 Introduction

Movie trailers can be found in abundance throughout the web using services such as video streaming platforms. However, not all types of content in trailers are suitable for every audience. Specifically, movie trailers may contain explicit, aggressive, or violent content that may be harmful to the psyche of young viewers. Previous research has documented that some of the negative effects of mass media in young viewers include aggression and anxiety (Wilson, 2008; Chang and Bushman, 2019), as well as increasing the risk of sexual onset and alcohol and drug consumption, unwanted pregnancies, and sexually transmitted diseases (Strasburger, 1989).

The Advertising Administration of the Motion Picture Association of America (“MPAA”) established guidelines for manually rating the age-suitability of movie trailers (Motion Picture Association). The rating of movie trailers is independent of the rating of the movie itself, as a trailer includes only a short overview of the entire movie. Due to the time consuming nature, as well as the challenges to scale the MPAA rating process, automating the task poses interesting challenges to multi-modal classification systems, as the source of the objectionable content can come from any, or the combination of, these sources: language (use of bad words or discussion of adult themes), images (graphic violent scenes, nudity, drug or alcohol use), and audio (loud noises and music score denoting suspenseful content). A successful rating approach should integrate evidence provided by the multiple modalities when making the predictions.

In this paper, we study the performance of different multi-modal deep learning methods, to automatically predict the MPAA age-suitability rating of movie trailers using cues from the video, audio, and text modalities. Our goal is to show the feasibility of automating the rating task, and in particular, the relevance of multimodal solutions. We explore the use of late fusion, feature concatenation fusion and Gated Multi-modal Unit (GMU) Fusion (Arevalo et al., 2017). Since the proposed pipeline does not use any type of metadata for trailers or movies, it can be easily extended to be applied to any type of online video content. The main contributions of this work are: (i) we introduce a new task in multi-modal classification; rating videos based on the MPAA rating metric for movie trailers; (ii) we introduce the Multi-modal Movie Trailer Rating (MM-Trailer) dataset that contains movie trailers and their corresponding MPAA tags, audio files, subtitles of the trailers, and the metadata of the target movie; and (iii) we demonstrate empirically that combining the different modalities yields significant improvements over the strongest monomodal model. Our results show that both, the GMU and late fusion approaches yield promising results.

2 Related Work

This work is related to four different areas, namely: (i) text classification, (ii) video classification, (iii)
audio classification, and (iv) movie classification datasets.

**Text Classification:** In (Martinez et al., 2019), the authors proposed an RNN-based architecture for detecting violence in movies on a segment level as well as the full movie level, by using the movie’s script. In Shafaei et al. (2019), the authors proposed an RNN-based architecture with an attention mechanism that jointly models the genre and the emotions in movie script to predict the MPAA rating of a full movie. The main difference between our work and aforementioned papers is that they only use scripts to predict the movie ratings (violence rating and MPAA ratings), while we employ various modalities (audio, video, and text) to predict if a trailer (not the entire movie) is appropriate or not for children. It should be noted that the rating schema is different for trailers compared to movies (details in Section 3), and movies are not freely available on the internet.

**Video Classification:** Early approaches, such as (Karpathy et al., 2014) on video classification using Deep Learning, explored the use of several temporal fusion methods for combining information from multiple consecutive video frames using features extracted from CNN architectures. The authors in (Donahue et al., 2015) introduced an end-to-end architecture based on a combination of CNNs used for feature extraction from RGB frames. The CNN features are then forwarded to an LSTM layer that models the temporal variation of frames. A different approach is followed in (Tran et al., 2015), namely 3D-CNN, where authors propose the use of a CNN variant that takes into account convolutions performed into both the spatial and temporal domains of a video. An expansion of the 3D-CNN approach was proposed by (Carreira and Zisserman, 2017), where the authors propose a two-stream 3D-CNN architecture for video classification. Again the two streams used as input RGB frame data and Optical flow images.

**Audio Classification:** In past research, several types of handcrafted feature extraction techniques have been proposed for the audio modality (Davis and Mermelstein, 1980; Geiger et al., 2013; Papakostas et al., 2017) with the ones being the most prominently used in the literature being Mel-frequency cepstral coefficients (MFCCs). However, recently several approaches have been proposed for combining audio features such as spectrogram information with deep learning architectures to perform audio classification (Papakostas et al., 2017; Hershey et al., 2017; Koutini et al., 2019). Audio has been explored as a modality for classifying movie content in several works such as (Rasheed and Shah, 2002; Hebbar et al., 2018). However, none of these methods has focused on the problem of movie trailer age-suitability rating.

**Movie Classification Datasets:** Several movie classification datasets have been proposed in the past. In (Demarty et al., 2014), the authors introduced MediaEval 2013 Violent Scene Detection, which provided annotations for detecting violent scenes in movies. In Constantin et al. (2020), the authors proposed an evaluation framework, for Violent Scenes Detection in Hollywood and YouTube videos along with a dataset (VSD96). Although these datasets are relevant to our work, they only cover the violence aspect and cannot address the problem of age-suitability rating (violence is only one of many aspects of age rating). In (Shafaei et al., 2019), the authors proposed a movie dataset focusing on the task of predicting the MPAA rating of the movie. However, the aforementioned dataset only includes movie scripts and corresponding metadata but does not include movie trailers or related age-suitability tags (As we mentioned earlier, the MPAA rating scheme is different for movies and trailer). In (Cascante-Bonilla et al., 2019), authors introduced Moviescope, a dataset for movie genre classification. Similarly, it does not include MPAA age-suitability rating labels for movie trailers.

### 3 Dataset

To the best of our knowledge, there is no previous trailer dataset with age-suitability rating. Thus, we assembled the multi-modal Movie Trailer dataset (MM-Trailer) \(^1\) by collecting the rated trailers from the IMDB website and YouTube. Typically trailers are advertising movies soon to be released and shown in theaters before a movie starts. Rating in trailers is shown by a colored band (red, yellow, green) and a message that appears at the beginning of the trailer. The rating of the trailers adheres to the rating of the movie being shown in the theater. For instance, if the movie playing at the theater is rated as NC-17 (no one under 17 is recommended to watch this movie), the green band trailer that is advertised before this movie may not be appropriate for children even if the color is green. The

---

\(^1\)https://ritual.uh.edu/RANLP2021/
yellow band is designed for trailers advertised on the internet, and it indicates that the corresponding trailer is suitable for “age-appropriate internet users” as visitors to sites are mainly adults. The last group of trailers are red band trailers; red color indicates the content is only appropriate for a “mature audience” or “restricted audience”.

Since our goal is to design an automated system that is able to predict which movie trailers are not recommended for children, we define only two classes of trailers for the dataset:

1. **Green-band trailers**: this category includes (i) trailers with the message “all audiences”, and (ii) green band trailers with “appropriate audience” whose associated movie is rated as G and PG.

2. **Red-band trailers**: all red-band trailers, these include restricted and mature audiences (not appropriate for children).

We also extracted separate audio files and trailer subtitles. Subtitles include narrator and actor speech. Some of the YouTube trailers include the video subtitle. For these cases, we pre-process the subtitles by removing timestamps to keep only words. For trailers that do not include a subtitle file, we use a python speech recognition tool (Zhang) to automatically generate the subtitle from the audio. Our dataset includes 11G of audio streams. For each trailer the audio file is a combination of background music and vocals together, so the duration of audio is the same as the duration of the trailer. The number of total words in all trailer scripts is equal to 1,478,139 (on average, there are 576 words per trailer). Note that 20,783 words of the vocabulary set are unique words. Table 1 shows the statistics of our dataset.

### Methodology

Our goal is to predict the age-suitability rating for movie trailers following the guidelines of the Advertising Administration of MPAA for trailer rating. The problem is formulated as a binary classification task where trailers are labeled as either appropriate for all audiences (green-band trailers) or restricted audiences (red-band trailers). To achieve this goal, the Multi-modal Movie Trailer Rating (MMTR) system is proposed. Within this system, the trailers are modeled as a fusion of three modalities: subtitles, audio, and video of the trailers. We train Recurrent Neural Networks (RNNs) for subtitles and audio, and a combination of Convolutional Neural Networks (CNNs) with LSTM for video, as separate streams in order to extract a representation for each respective modality. Then, we combine all stream representations using a fusion module to take advantage of the cues coming from different modalities. Figure 1 shows the overall design for the system architecture.

Our approach is based on independently identifying the best individual modality model and then combining information from all three monomodal models (subtitle, audio, and video) through one of the following three fusion methods: (i) Gated Multi-modal Unit (GMU) (Arevalo et al., 2017), (ii) Feature Concatenation Fusion, or (iii) Late Fusion. All modules of the system are described in the following sections.

#### 4.1 Text Stream

The subtitles of the trailers are a rich source of information. They can help in identifying the topic of the video content. Moreover, the presence of specific words in the dialogue can be a strong indicator for some types of sensitive content, while more subtle cues can be inferred from analyzing the entire transcript. To model the information originating from the subtitles, we feed them to the following modules:

**BERT + Long Short-Term Memory (LSTM) with Attention**: We use BERT (Devlin et al., 2018) to leverage the well-known power of transformer-based word representations. The word vectors are then passed to an LSTM layer to model the sequence of the words in order to extract the semantic information of the text. Afterwards, the resulting hidden representation of the LSTM is passed to an attention mechanism (Bahdanau et al., 2014) to find the importance of each word in the dialogue. Even though BERT has seen a series of improvements (RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019)), our goal in this paper is to present empirical evidence that a multi-modal approach can solve this task with acceptable performance, the specific contextualized embeddings used being of less relevance.

**Emotion Vector**: We expect to observe that
strong negative emotions (fear, anger, sadness) correlate more with red band trailers. Similarly, positive emotions, such as joy, are more correlated with green band trailers. We made this assumption following research by (Shafaei et al., 2019) where they found promising results for using emotions in a movie rating task.

We model emotions with the use of the DeepMoji model (Felbo et al., 2017). This model was trained using 1.2 billion tweets with emojis to understand how language is used to express emotions. Recent work in abusive language detection shows promising results from using DeepMoji (Safi Samghabadi et al., 2019), thus it seems reasonable to expect good results in this task as well. To incorporate this model into our system, the last hidden layer representation of the pretrained model was used to transfer the text to emotional feature vectors. Finally, the emotion vector was concatenated with the output of the attention and the entire vector was passed to a fully connected layer to further fine-tune the joint representation.

4.2 Video Stream

The video modality is a rich source of visual and temporal cues that are useful for analyzing multimedia content. Specifically, in this task, video can help for modeling the objectionable content such as a depiction of nudity or bloody scenes and suggestive elements. To this end, in order to learn spatiotemporal video features, a CNN-LSTM model based on the works by (Donahue et al., 2015) and (Yue-Hei Ng et al., 2015) is adopted. Each video is sub-sampled to a fixed number of frames, evenly distributed across its duration to form a visual temporal sequence. The raw RGB frames are used as input to a CNN model. This CNN model produces a feature representation for spatial information within each frame. The output of the final pooling layer of the CNN is passed to an LSTM that models temporal dependencies between frames.

4.3 Audio Stream

The audio of the trailer can help the model to learn the genre and theme of the movie, and as a result, it is a powerful tool to distinguish red-band trailers from green-band ones. For example, horror and thriller movies (that usually include suspenseful music) are less likely to be suited for children. In addition to the music score, the emotion conveyed by the speakers’ tone and pitch can provide relevant cues for rating the trailer. It should be noted that the entire audio is used in our model (the music and dialogue combined). To model the audio, the Mel Frequency Cepstral Coefficients (MFCC) are extracted from the audio stream. MFCCs are one of...
the most common feature representations for audio classification (Andén and Mallat, 2011) and speech recognition tasks (Tiwari, 2010). The entire audio is divided to n chunks, $n \in \{10, 20, 50, 100\}$, then the MFCC feature vector is extracted for each chunk. Moreover, by performing averaging over the MFCC vector in each chunk, a fixed-length representation for the entire audio, regardless of its duration is obtained. The vector is then passed to an LSTM module to model the MFCC variations during the entire video. Lastly, by adding an attention mechanism, the model learns the importance of each audio chunk and feeds the weighted average of LSTM hidden representation to a fully connected layer that helps the model to be fine-tuned for the task.

4.4 Fusion

The goal of the fusion module is to learn to predict the rating of the trailer by integrating evidence from the video, audio and text modalities. We evaluate three established fusion methods in order to form a unified representation for each trailer.

Gated Multi-modal Unit (GMU): The GMU allows the model to learn an intermediate representation by combining the different modalities, where the gate neurons learn to decide the contribution of each modality to the intermediate representation. A great advantage of the GMU model is its ability to adjust the activation from each modality depending on the specific instance. This method is inspired by control flow in recurrent architectures. In RNN models, the recurrent units decide how much the current and previous evidence engage in building the current state. In GMUs, the activation function for building the output using different modalities is measured, in order to form a unified intermediate representation for all modalities.

The original GMU was successfully applied to a movie dataset of plot synopsis and movie posters to predict genre. In the original paper, the authors implemented a bimodal system (the equation is provided in the Appendix). We follow their formulation to extend the model to include three modalities using the straightforward approach discussed in their paper. The exact formulation is shown in Equation 1; where $W_i$, $Y_i$ are learnable parameters, $x_i$ is the feature vector for modality $i$ and $[.,.]$ stands for concatenation.

$$h_1 = \tanh(W_1x_1)$$
$$h_2 = \tanh(W_2x_2)$$
$$h_3 = \tanh(W_3x_3)$$
$$z_1 = \sigma(Y_1,[x_1,x_2,x_3]) \quad (1)$$
$$z_2 = \sigma(Y_2,[x_1,x_2,x_3])$$
$$z_3 = \sigma(Y_3,[x_1,x_2,x_3])$$
$$h = z_1 * h_1 + z_2 * h_2 + z_3 * h_3$$

Feature Concatenation Fusion: One popular fusion method is generating a joint multi-modal representation through feature concatenation (Baltrušaitis et al., 2018) where the representation vectors of each modality are concatenated, and the unified representation is passed through multiple hidden layers or used directly for the prediction.

Late Fusion: Another vastly used fusion method is late fusion (Fu et al., 2015). In late fusion, different modalities are merged in the decision level using various rules (e.g., majority voting, averaging) (Baltrušaitis et al., 2018). Here, the average of all modality outputs is calculated and used as the final output.

Before performing either feature concatenation or GMU based fusion, information from each modality is represented with a feature vector extracted from pretrained models, acting as modality streams. Then, we transform the vectors from all modalities into a single vector using the GMU or concatenation module. Finally, we pass the fused representation to a fully connected layer, creating a vector of size two (we have two classes). The sigmoid function is then applied to the two-dimensional vector to assign a label to each trailer. For late fusion, we capture the output of each single modality model before the sigmoid function (vectors of size two) and compute the average. Lastly, we pass the single representation to a sigmoid function for the classification.

5 Experiments

The goal of this section is to demonstrate that a multi-modal approach is an effective way to solve the task. We, therefore, compare the prediction performance of single modality models against all multimodal variations of the system.

As mentioned in the dataset section, the MM-Trailer dataset is imbalanced. Thus, to obtain reliable results, 5 fold cross-validation was selected as an evaluation method. In each fold, we select 10%
of the train sent as the validation set to obtain the best model. It should be noted that the dataset was split using the stratified approach, so as to ensure that each set has the same proportion of examples from each class. The metric used to evaluate the performance is the weighted F1 score, averaged over all 5 folds for each experiment.

5.1 Baseline Methods

Most Frequent Baseline: The first baseline is a naive approach to show that the problem is not easy to solve. In this model, we assign the most frequent class to all the instances in the validation and test sets, and we measure the F1 score by considering the ground truth label.

Text Baseline - Traditional Machine Learning: For the text baseline model, we provide a traditional machine learning method with hand-crafted features. We extract unigram and bigram features from subtitles and apply term frequency-inverse document frequency (TF-IDF) as the weighting scheme. Then, the feature vectors are passed to an SVM model for classification. We chose an SVM model as it performed well on the similar task of violence detection (Martinez et al., 2019).

Text Baseline - BERT + Attention + NRC: A popular resource to extract the emotion in the text is the NRC emotion lexicon (Mohammad, 2011). This dictionary maps words to eight different emotions (anger, anticipation, joy, trust, disgust, sadness, surprise, and fear) and two sentiments (positive and negative). Using this dictionary, we compute the normalized count of words per emotion over the entire subtitle and create a vector of size 10 for each trailer. We use this vector as an alternative to DeepMoji vector in the model.

Text Baseline - DeepMoji + fully connected layer: To show how much emotion by itself can contribute to the prediction of rating, we only use the DeepMoji vector as the input and pass it to a fully connected layer and sigmoid classifier for the prediction.

Video Baseline: Our video baseline is based on the deep 3-dimensional convolutional network (3D CNN) architecture proposed by (Tran et al., 2015). The 3D-CNN architecture applies 3D convolution and 3D pooling operations on video volumes instead of images. Each video is sub-sampled to an 18 evenly distributed frames that are used as input to the model. The training was performed for 50 epochs, using a 0.5 dropout rate, with a learning rate of $10^{-5}$ and a batch size of eight samples.

Audio Baseline: CNNs have shown promising results for audio classification (Hershey et al., 2017). To this end, for each full video, the log-Mel spectrogram is extracted from the audio using the LibROSA python library (McFee et al., 2015) and then used as input to a CNN architecture. For the log-Mel spectrograms 128 Mel-spaced frequency bins were used, while for the CNN model for this baseline, Inception V3 was adopted. The CNN model was trained for 100 epochs using a batch size of 64 samples and a learning rate of $10^{-5}$. An early stopping policy was used during training to avoid over-fitting.

6 Results

Table 2 summarizes the results of our experiments. To examine the contribution of each modality for the rating task, we report the results for all single modality models; Audio only Model (A-MFCC), Text only Model with DeepMoji (T-BAD), and Video only Model (V-CNN/LSTM). As expected, our experimental results confirm that by leveraging all modalities we achieve a better result. As noted in Table 2 the highest weighted F1 score, 86.06%, is achieved by the GMU Fusion variant of the MMTR model with all modalities. This result improves the weighted F1-score of the best single modality model (T-BAD) over 3 percentage points ($P < 0.05$ based on t-test).

We also report the result for different combinations of two modalities using all fusion methods to show the effect of engaging all modalities (T-BAD + A-MFCC, T-BAD + V-CNN/LSTM, A-MFCC +V-CNN/LSTM, T-BAD + A-MFCC + V-CNN/LSTM). Based on the results, the combination of two modalities works better than every single modality, yet not as good as the combination of all modalities.

When comparing the different fusion approaches, we can see that GMU fusion outperforms the concatenation fusion systems. We speculate that the gains from GMU come from the ability of the gated unit to dynamically adapt the contribution of each modality to the intermediate representation. Statistical significance testing using t-test, demonstrated a significant difference between GMU and feature concatenation fusion ($p−value < 0.05$). However, the test does not confirm a significant difference.
Table 2: Evaluation of the different variants of the MMTR system and other baselines using the MM-Trailer dataset. WF stands for weighted F1 score and results are averaged over 5 folds. A '*' indicates that the difference between the two classifiers’ performance is shown to be statistically significant.

<table>
<thead>
<tr>
<th>Model</th>
<th>Test-WF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most Frequent Baseline</td>
<td>60.37</td>
</tr>
<tr>
<td>Text Baseline - Traditional Machine Learning</td>
<td>75.02</td>
</tr>
<tr>
<td>Text Baseline - BERT+ Attention (T-BA)</td>
<td>81.99</td>
</tr>
<tr>
<td>Text Baseline - BERT+ Attention + NRC</td>
<td>81.67</td>
</tr>
<tr>
<td>Text Baseline - DeepMoji + FC</td>
<td>68.23</td>
</tr>
<tr>
<td>Video Baseline</td>
<td>75.33</td>
</tr>
<tr>
<td>Audio Baseline</td>
<td>72.62</td>
</tr>
<tr>
<td>Audio- MFCC (A-MFCC)</td>
<td>73.86</td>
</tr>
<tr>
<td>Text- BERT+ Attention+ DeepMoji (T-BAD)</td>
<td>82.67*</td>
</tr>
<tr>
<td>Video- CNN/LSTM (V-CNN/LSTM)</td>
<td>79.41</td>
</tr>
<tr>
<td>T-BAD + A-MFCC</td>
<td>82.41</td>
</tr>
<tr>
<td>T-BAD + V-CNN/LSTM</td>
<td>84.12</td>
</tr>
<tr>
<td>A-MFCC + V-CNN/LSTM</td>
<td>79.68</td>
</tr>
<tr>
<td>T-BAD + A-MFCC</td>
<td>82.17</td>
</tr>
<tr>
<td>T-BAD + V-CNN/LSTM</td>
<td>82.80</td>
</tr>
<tr>
<td>A-MFCC + V-CNN/LSTM</td>
<td>78.70</td>
</tr>
<tr>
<td>T-BAD + A-MFCC</td>
<td>83.37</td>
</tr>
<tr>
<td>T-BAD + V-CNN/LSTM</td>
<td>83.34</td>
</tr>
<tr>
<td>A-MFCC + V-CNN/LSTM</td>
<td>80.35</td>
</tr>
<tr>
<td>Late (Fusion using two modalities)</td>
<td>85.60</td>
</tr>
<tr>
<td>Mid (Concat) (T-BAD + A-MFCC + V-CNN/LSTM)</td>
<td>82.75</td>
</tr>
<tr>
<td>Fusion using all Modalities (MMTR)</td>
<td>86.06*</td>
</tr>
<tr>
<td>Late (Fusion using two modalities)</td>
<td>82.41</td>
</tr>
<tr>
<td>Mid (Concat) (T-BAD + A-MFCC + V-CNN/LSTM)</td>
<td>84.12</td>
</tr>
<tr>
<td>Fusion using all Modalities (MMTR)</td>
<td>79.41</td>
</tr>
<tr>
<td>Early (Fusion using two modalities)</td>
<td>73.86</td>
</tr>
<tr>
<td>Mid (GMU) T-BAD + A-MFCC + V-CNN/LSTM</td>
<td>82.75</td>
</tr>
<tr>
<td>Fusion using all Modalities (MMTR)</td>
<td>86.06*</td>
</tr>
</tbody>
</table>

between late fusion and GMU. Thus, we can claim that for the trailer age-suitability problem, late fusion can generalize as good as GMU fusion.

The results for T-BAD and T-BA indicate that DeepMoji is a relevant feature for the rating task, and it helps the model to better discriminate red-band trailers from green-band ones. However, the result of DeepMoji+FC shows that the DeepMoji model is not sufficient to solve the task.

To obtain a better understanding of fusion results, we also provide other evaluation metrics using the MMTR system variant with GMU Fusion in Table 3 (as GMU version is the winner approach based on the result table). Based on the detailed result, most of the incorrectly predicted instances are red-band trailers. The first potential explanation behind this observation is that there are fewer instances of red-band trailers in our training set compared to green-band. As a result, it is more difficult for the model to capture all patterns in this class. The second reason may be the diversity of video content in red-band trailers. Recall that this class covers any content that is not appropriate for children.

It is thus reasonable to assume that this class is more heterogeneous than the green band class. We plan to explore the possibility of a fine-grained classification of objectionable content as the next steps in this work.

7 Discussion

To analyze the weaknesses and strengths of the MMTR system, we first investigate the incorrectly predicted cases using the most effective version of the system (GMU Fusion) on each fold of the data. By averaging results over all folds, about 35% of incorrectly predicted cases with the MMTR system are also incorrectly predicted by each and every modality independently, fusion is thus unlikely to help in this case. We found that in about 50% of the instances where two modalities predict the wrong rating, the MMTR system justifiably trusts the single modality that is correct. In about 93% of the cases where only one modality is wrong, the MMTR GMU Fusion variant predicts the correct label, relying on the other two modalities.

After averaging results among all folds, we no-
that the MMTR GMU Fusion variant system is not able to predict about 38 out of 294 instances per fold. We watched 40 incorrectly classified trailers (selected across all folds) to analyze why the model is not able to successfully predict the label. We introduce the following hypothesis for each of the individual modalities:

1) **Text Modality:** One main source of errors in text modality comes from the output of the speech recognition tool. First, the free version of the tool only works on short audio files. As a result, we split the whole audio to 10-second chunks. Thus, it is possible that we miss some words if the audio is cut off in the middle of the word. Also, low-quality audio impacts the word recognition rate of the automatic speech recognition system, which in turn cause the model not to recognize the specific bad words present in the video or the other way around, generate bad words by mistake (detect “please” as “pussy”). However, in some cases, the trailer either has very little speech (less than 10 words) or there is really no sensitive content in the language used. Not surprisingly, the text modality cannot work properly. Finally, in some green band videos, we observed that the trailer subtitles have the words “gun” and “shot”, thus they are predicted incorrectly by the text modality. It seems that the text model is biased against the occurrence of these words that are presumably strongly correlated with violent content.

2) **Video Modality:** One main reason that the video modality model misses the sensitive content may relate to the video sampling rate. The inappropriate/violent scenes in these trailers disappear fast, or they may appear with a low frequency. As a result, we may miss them during sampling the frames in our model. The second potential reason is the quality of the trailers. We recognized that some of the trailers are old or are available in small files, so the frames are blurry, and even in some cases, not very clear to the human viewer. Lastly, we found out, there are some green-band trailers that still include brief sensitive content like the depiction of guns and blood, and our video modality model predicts them as red-band. These instances are mostly the R-rated movies that are sanitized for the trailer. However, the theme of the movie reflects itself in some frames. We can conclude that sometimes a single rating is not sufficient for expressing the type of the content, and as future work, we can predict a list of sensitive material in the video instead of a single label.

3) **Audio Modality:** In some cases, the music of the trailer is not compatible with the content. For example, we encountered musical movies with a high level of violence, but with smooth jazz music. Thus, it is difficult for the audio modality to distinguish between appropriate and inappropriate content. Moreover, in audio modality (similarly to the video modality), we capture samples from the continuous stream. Hence, if the intense audio (such as a scream or a gunshot) happens in a short period, our model may miss that.

We also investigated the genre of incorrectly predicted trailers in one of the data folds. The interesting point is that, for incorrectly classified red-band trailers, 55% are categorized as “Thriller” or “Horror” movies and 30% as “Comedy” (based on IMDB metadata). We do not incorporate metadata into our model to make the model suitable for any kind of online content. This observation shows that the genre of the movie can be a potential feature for the model if we have metadata available.

**8 Conclusion**

In this paper, we present a deep learning system named MMTR for automating the task of movie trailer age-suitability rating. MMTR fuses information from the video, audio, and text modalities. We also introduce a new data set to support research in this area. This dataset contains movie trailer videos along with their rating and metadata. The results of comparing our model with strong baselines demonstrated that the task is not easy, and a complicated multi-modal systems (GMU and late fusion) can achieve performance gains compared to other baselines. Beyond the practical use of a binary classification system, we are interested to move to the more challenging task of detecting the type of objectionable content and introducing explainability elements within the MMTR System.
Acknowledgments

This work was partially funded by the National Science Foundation under award 2036368.

References


François Chollet et al. 2015. Keras. https://keras.io.


Jürgen T Geiger, Björn Schuller, and Gerhard Rigoll. 2013. Large-scale audio feature extraction and svm for acoustic scene classification. In 2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, pages 1–4. IEEE.


A Appendices

A.1 Implementation Details

This section discusses how the different streams of the MMTR system were implemented. For all experiments, we used the ADAM optimizer and the Cross-Entropy Loss function. Hyperparameter values were selected using manual tuning of the model using the best value of the weighted F1 Score for the validation partition, as the criterion.

Text Stream: For the BERT model we used the implementation provided by (Wolf et al., 2019). The LSTM layer consisted of 256 RNN units. Training was performed for 50 epochs, using a 0.3 dropout rate, with a learning rate of $10^{-5}$ and a batch size of eight samples.

Video Stream: For each movie trailer, frames were extracted with a rate of one frame per second, from which 18 evenly distributed frames were used to represent each video within the model. For the CNN feature extractor, we used the Inception V3 architecture pre-trained with ImageNet (Russakovsky et al., 2015). The model was trained using a learning rate of $10^{-5}$ and a batch size of 64 samples and by using an early stopping policy to avoid overfitting.

Audio Stream: For each trailer, the audio was split in 20 chunks. For the LSTM layer 256 RNN units were used. Training was performed for 50 epochs, using a 0.1 dropout rate, with a learning rate of $10^{-5}$ and a batch size of eight samples.

System Specifications: All models were developed using the Tensorflow (Abadi et al., 2015), Keras (Chollet et al., 2015) and PyTorch (Paszke et al., 2019) libraries on a machine with Ubuntu 14.04 LTS as the operating system. The system had an Intel Core™ i7 CPU running at 2.67GHz with four cores and 8 GB RAM memory. The video card used was a GeForce GTX 1080 Ti.

A.2 The GMU model:

The original equation of the GMU is represented in Equation 2; where $W_v$, $W_t$, and $W_z$ are learnable parameters, $x_v$ and $x_t$ are modality feature vectors and $[., .]$ stands for concatenation.

$$
h_v = \tanh(W_v x_v), h_t = \tanh(W_t x_t) 
$$

$$
z = \sigma(W_z [x_v, x_t]) 
$$

$$
h = z \ast h_v + (1 - z) \ast h_t
$$

Note that in the extension to more than two modalities, the model ends up having more parameters as the the gates are no longer tied. But as shown empirically, this does not seem to be a problem for the model.
A Domain-Independent Holistic Approach to Deception Detection

Sadat Shahriar
University of Houston, Texas, USA
sshahriar@uh.edu

Arjun Mukherjee
University of Houston, Texas, USA
arjun@cs.uh.edu

Omprakash Gnawali
University of Houston, Texas, USA
gnawali@cs.uh.edu

Abstract

The deception in the text can be of different forms in different domains, including fake news, rumor tweets, and spam emails. Irrespective of the domain, the main intent of the deceptive text is to deceive the reader. Although domain-specific deception detection exists, domain-independent deception detection can provide a holistic picture, which can be crucial to understand how deception occurs in the text. In this paper, we detect deception in a domain-independent setting using deep learning architectures. Our method outperforms the State-of-the-Art (SOTA) performance of most benchmark datasets with an overall accuracy of 93.42% and F1-Score of 93.22%. The domain-independent training allows us to capture subtler nuances of deceptive writing style. Furthermore, we analyze how much in-domain data may be helpful to accurately detect deception, especially for the cases where data may not be readily available to train. Our results and analysis indicate that there may be a universal pattern of deception lying in-between the text independent of the domain, which can create a novel area of research and open up new avenues in the field of deception detection.

1 Introduction

In the current era of the flood of information, deception has become an undeniable event, causing a financial or political catastrophe and even the loss of human lives. Often, we do not have the necessary resources to validate a tweet or a catchy social media forwarded news link. Our idea is to capture the writing style hidden “between-the-lines”, intended to deceive the reader. The deception can be in the form of any textual stream and on any topic. So, our objective is to find a holistic model that can leverage thousands of textual resources and find a learning architecture to decode the deception.

Adopting the definition of Deception from Burgoon and Buller, we define Deceptive Text as any textual content that aims to misconstrue an affair in a deliberate way causing the reader at a disadvantage either directly or indirectly (Burgoon and Buller, 1994). Deceptive text can be of various forms. For example, in the news and public media domain, the deceptive text is known as Fake News. In social media, a deceptive text can happen in the form of a rumor. Spam or a phishing email is treated as deceptive content in the personal mail or messaging domain. Each domain’s deceptive text has distinct ways to deceive the reader. While fake news can spread falsified propaganda, spam or phishing email can be used for merely monetary gain. Therefore, the ways of formulating a deceptive text can have significant variations. Notwithstanding, all deceptive texts share a common goal of tricking the reader and thus, a general deception pattern should exist in these texts. Unraveling the pattern can play a pivotal role to provide a holistic view of deception, which in-turn can bolster the deception-detection. However, Gröndahl and Asokan indicated that existing works fail to generalize the deception across different domains (Gröndahl and Asokan, 2019). In this work, we hypothesize that— (H1) A deep learning architecture trained on a generalized deceptive-writing setting (both in-domain and out-of-domain data) can better understand the underlying general pattern of deception than using the in-domain data only.

On the rise of a fairly new event, we may not have data available to detect deceptive text floating around social and mass media, especially the labeled data for supervised training. Such occasions pose a unique challenge to stop the spread of misinformation. A holistic deception detection system can come in handy in such situations. For example, although the pandemic was as ancient as human civilization, in the age of massive data availability,
COVID-19 becomes a new event, and misinformation caused by this event can be hard to battle. Therefore, we hypothesize that (H2) A generalized dataset can be helpful to detect deception in a new event, even when little or no in-domain data is available. To test our two hypotheses, we train and fine-tune a BERT model, SBERT model, and character-level-CNN model (Devlin et al., 2019; Reimers and Gurevych, 2020). We further investigate the intermediate-layer learning mechanism using t-SNE visualization and attention-weight analysis.

Researchers worked meticulously to model deception on different domains, but a unified approach has not been successful. Hernández-Castañeda et al. suggested a cross-domain approach for a generalized deception-detection model (Hernández-Castañeda et al., 2017). Although they claimed to build a domain-independent system, their choice of the datasets, namely DeRev, OpSpam, and Opinion (a dataset where the participants were told to lie about their opinion about selected topics), are of the type opinion-only. Moreover, the Opinion dataset can hardly be treated as deceptive. That is because when people are told to lie about an opinion, the lie may not have the potential of deceiving someone which contradicts our definition of Deception. On the contrary, our choice of datasets have at least three variations of categories – fake news, Twitter rumor, and spam.

Thus, we summarize the main contributions of this paper as–

• Our work is the first to propose a domain-independent holistic approach to detecting deception leveraging available public datasets.
• We quantitatively show that for an unseen event, only a fraction of the total available data can be helpful in successfully detecting the deception.

2 Literature Review

In this research, we aim to detect deceptive content intended to mislead people rather than entertain them. Therefore, the deceptive contents can be viewed as deceptive news, disinformation, cherry-picking, and click-bait (Zhou and Zafarani, 2020). There have been several approaches for manual fact-checking, both in the form of expert-based (Pol, accessed February 2, 2021; gos, accessed February 2, 2021; fac, accessed February 2, 2021) and crowd-sourced (CRE, accessed February 2, 2021). However, given the enormous influx of information, such manual approaches are time-consuming and often biased. Therefore, automated fact-checking came in handy. Based on how users spread the falsified information, researchers adopted following approaches to detect deceptive content: news-cascade, which is a tree-like structure to analyse the propagation in social-media (Ma et al., 2018), and Propagation Graph(Zhou et al., 2019b). However, such approaches are also constrained by the availability of propagation detection resources. Moreover, false information cannot be detected before it spreads out. Additionally, some research tries to detect false news based on the source credibility (Viviani and Pasi, 2017). Nevertheless, the stream of new sources now and then makes the task challenging.

Therefore an AI-based method aiming to detect deception based on the textual content can be handy. Because of the fewer dependencies and availability of content, many researchers worked in that direction. Zhou et al. divided the task into two steps: (i) how well the deceptive news content is captured, and (ii) how well the classification model performs to detect deception (Zhou and Zafarani, 2020). Approaches, such as the Bag of Words (BOW) model, POS tagging, rhetorical relationships, were used as features to detect deceptive news (Bhatt et al., 2018; Zhou et al., 2019a; Zhou and Zafarani, 2020). Nevertheless, we are more interested in the semantics, as the task of deception may lie in between the text. Word level context embeddings, including Continuous Bag of Words (CBOW) and skip-gram models were used to represent text for detecting fake news (Potthast et al., 2017). Along with such representations, several machine learning algorithms are used for classification purposes. Additionally, with the rise of deep learning, Convolutional Neural Network (CNN) and many variants of Recurrent Neural Network (RNN) are used as well (Li et al., 2019; Ajao et al., 2018).

Recently the Bidirectional Encoder Representations from Transformers (BERT) model and its variants have gained enormous popularity due to its pretraining capability (Devlin et al., 2018). Müller et al. proposed COVID-Twitter-BERT (CT-BERT), which is pretrained on COVID-19 related Twitter messages (Müller et al., 2020). Such task-specific BERT model outperforms the generic BERT models in the significant margin in many COVID-
related classification tasks, including AAAI2021 COVID-19 shared challenge of COVID-19 Fake News Detection (Glazkova et al., 2021). A similar fine-tuning approach is used by Liu et al., where they proposed a two-stage approach for short fake news detection. Unlike the original BERT model, they utilized all hidden states to apply the attention mechanism to calculate weights for text representation. Their approach produced a 34% accuracy in the LIAR dataset (Liu et al., 2019). Kaliyar et al. proposed FakeBERT, which uses a CNN model after the BERT embedding layer (Kaliyar et al., 2021).

Although the current deception detection methods work well, the methods are highly dependent on the training of the specific domain. On the contrary, we intend to eliminate dataset-specific training and train our model for the generic deception detection task.

3 Dataset

We curate ten datasets for our analysis. We broadly categorize them as—i) Spam, ii) Fake News, and iii) Rumour. The details of the datasets are described below.

3.1 Email and Text Spam

For the spam datasets, we select two personal-messaging datasets. First, we select SMS Spam collection Dataset from UCI Machine Learning Repository (Almeida et al., 2011). This dataset collection has messages collected from different sources totaling 5,574 messages, of which 4,827 are Hams, and 747 are Spams. We also curated the Enron-Spam datasets, which is a benchmark dataset for email spam collection from six different users (Metsis et al., 2006). There were 15,421 Spam emails and 14,923 Ham emails, totaling 30,344 emails.

3.2 Fake News

For the Fake News datasets, we start by collecting the COVID-19 related fake news. The first one is Constraint@AAAI2021 - COVID19 Fake News Detection in English (Patwa et al., 2021). The data are collected from various social media platforms. The training data has 6,420 texts, validation data has 2,410 texts, and the test data has 2,140 texts. The dataset has overall 52% real and 48% fake news. Another COVID-19 related dataset we use is Zenodo– COVID Fake News Dataset (Banik, 2020). The Zenodo COVID dataset has 10,201 texts, out of which there are 9,733 fake news and only 468 real news.

Next, we collect a dataset of varying unreliability, developed by Rashkin et al., where each text was considered as either a Satire, a Hoax or a Propaganda (Rashkin et al., 2017). Unlike our definition of Deceptive-text, Satire cues the reader of the news being a joke only, and thus, we treat Satire as a non-deceptive text. The Hoax and the Propaganda are meant to misguide people, and therefore, we treat them as Deceptive-text. There are 38,859 texts, of which 24,839 were deceptive texts and 14,020 non-deceptive texts. Additionally, they collected 4,362 data from PolitiFact, which are rated in a 6 pt. scale, namely, True, Mostly-True, Half-True, Mostly-False, False, Pants-on-Fire False. We consider the first three as non-deceptive text, and the last three are deceptive text. The dataset comes with a separate train, test, and a dev set.

Additionally, we use the FakeNewsNet dataset which comes with real and fake news content from PolitiFact and GossipCop (Shu et al., 2018, 2017a,b). In total, there were 23,196 data. The last dataset we use in the Fake News Category is the Liar benchmark dataset (Wang, 2017). Along with the text and the labels, the dataset comes with 12 other metadata. The dataset comes with a separate train, test, and a dev set. In total, the dataset contains 12,791 texts.

3.3 Rumour

We collect the PHEME dataset of rumors and non-rumors, which contains the Twitter rumor and non-rumors during breaking news, namely in the events of Charlie Hebdo, Ferguson, Germanwings, Ottawa Shooting, Sydney Siege (Zubiaga et al., 2016). We treat rumors as deceptive text and non-rumors as non-deceptive text. In total, we have 6,425 texts.

4 Methodology

4.1 Deep Learning Frameworks

4.1.1 Bidirectional Encoder Representations from Transformers (BERT) model

BERT is a pre-trained language representation model proposed by Devlin et al. (Devlin et al., 2019). BERT is trained on a bidirectional setting of context and with the objective of Masked Language Modelling and Next Sentence Prediction. The transfer learning capability of BERT makes it a popular candidate for many NLP tasks, such as sentiment classification, fake news detection, question-
answering. The BERT model consists of several transformer blocks, which are made of attention and feed-forward layers (Vaswani et al., 2017). In this work, we fine-tune the bert-base-uncased version of the BERT model, which consists of twelve transformer blocks. The 768-dimension output vector from the BERT model (position of \([CLS]\) token) is fed to a one-layer fully-connected network for classification. We use the recommended batch size of 16, and other hyperparameters (epoch, hidden-unit, learning-decay-rate) are chosen by cross-validation.

4.1.2 Sentence-BERT
The Sentence-BERT (SBERT) is a modified version of BERT capable of representing semantically meaningful sentence embedding (Reimers and Gurevych, 2019). SBERT is based on siamese and triplet networks for fine-tuning over BERT. It performs a pooling operation (min, max, or mean pooling) on the output of BERT. SBERT has a much faster running time compared to BERT. In our work, we use the pre-trained SBERT model and fine-tune it with two fully-connected hidden layers on top of that. We use cross-validation to choose hyperparameters (hidden units, batch size, and epochs).

4.1.3 Character-level-CNN model
Convolutional Neural Net (CNN) is a popular network of computer vision tasks, and it extends to NLP tasks (Kim, 2014). The Character-Level CNN (Char-CNN) was first proposed by Zhang et al., which is capable of dealing with Out-Of-Vocabulary (OOV) words by focusing on the character-level rather than the word or sentence level (Zhang et al., 2015). The Char-CNN consists of six convolutional layers and three fully-connected layers, followed by a max-pooling layer. We empirically choose the convolution filters to be 256. The fully-connected layer units, batch size, and the dropout rate is chosen by cross-validation.

4.1.4 Ensemble model
There can be two ways to ensemble the DL models—1. Hard Decision and 2. Soft Decision. In Hard Decision, we make the prediction based on the majority voting on a test sample. However, the majority voting can eliminate the effect of a strong probability confidence model by predicting a class even when two of the three models predict a class with weak probability confidence. Therefore, in the Soft Decision ensemble, we take the average softmax probability score of the DL models before the prediction phase. For illustration, let’s assume in a two-class classification setting, the softmax layer output of BERT, SBERT, and Char-CNN is \([b_0, b_1], [s_0, s_1], [c_0, c_1]\) respectively. The ensemble model will have the softmax probability output of, \([b_0 + s_0 + c_0, b_1 + s_1 + c_1]\). In this work, we use the Soft Decision ensemble model.

4.2 Experimental Set-up
We use the dataset-provided test set for Liar, Rashkin-Politifact, and COVID-AAAI datasets for the general holistic deception detection task. For the rest seven datasets, we randomly sample the data into training, validation, and test set as 60%-20%-20% and repeat the experiments for three different splits. We report the average performances of the three splits.

For the new-event holistic deception detection task, we select COVID-19 as a new event and combine the test set of COVID-AAAI and COVID-Zenodo. First, we train only on the out-of-domain eight datasets. Then, we add 20%, 40%, 60%, 80%, and 100% in-domain COVID data along with the out-of-domain datasets for training. We also use 20% of the training data as the validation set. This setup enables us to examine the strength of a holistic model in an unknown or little-known event.

5 Results and Discussion
To test the hypotheses, we divide our experiments in two parts: i) General Holistic Deception Detection and ii) New-Event Deception Detection.

5.1 General Holistic Deception Detection
The idea behind a general holistic deception detection task is to build a generalized system that will be able to detect deception in the text irrespective of the topic or target domain. Being a domain-independent system, the holistic model may have more robustness than the domain-specific model.

From the standalone model performances on Table 1, we can see the varying degree of performance across different datasets. However, in almost every case, BERT outperforms the Char-CNN and SBERT model. With the self-attention mechanism, BERT may better capture the nuances within the text than the Char-CNN model, which can give potential cues to detect deception. Despite being a variant of BERT, the SBERT model may lose some
The superior performance of the BERT model
outperforms the other models. However, current
SOTA accuracy on the COVID-AAAI dataset is
achieved using a COVID-twitter-BERT model,
which was trained on a large corpus of COVID-
related tweets, and outperforms our best model by
3.30% (Glazkova et al., 2020). As explained later,
with the cost of adding more in-domain data, the
performance tends to improve. Thus, we may have
achieved a better score if we would have trained on
more in-domain data.

For the ENRON email spam dataset, our ensemble
model performs the best amongst all the models.
It outperforms the SOTA hybrid network for spam
email detection by 3.70% in F1-Score (Douzi et al.,
2020). Our ensemble model achieves the best accuracy in the
SMS Spam dataset, which outperforms our best model by
0.78% (Almeida et al., 2011).

The soft decision ensemble model does not perform
better on five of the ten datasets than the standalone
datasets. As the ensemble model takes an
average of the softmax decision of the models, a
weak classifier gets an equal weight to a strong
classifier, which in turn may hurt the final decision.
Further investigation may be undertaken to develop
a weighted average ensemble of the models for a
more robust classifier.

The superior performance of the BERT model
information by the fine-tuning and the pooling pro-
cess, which creates further research direction to-
towards fine-tuning the SBERT model for deception
detection. We find the best overall performance for
the ensemble model, with an accuracy of 93.42%,
and an F1-Score of 93.22%. The best performing
standalone model – BERT lags slightly behind that
with an accuracy of 92.72%, and an F1-Score of
92.50%.

For the PHEME dataset, we find the best performing F1-Score in the ensemble model as
82.74%, which is better than the current top scorer
staA-HitPLAN based model (77.4%) (Khoo et al.,
2020). Similarly, for the Liar dataset, the ensemble
model achieves the best performance with an accuracy of 70.72%, and F1-Score of 62.60%, which
performs the other models. However, current
SOTA accuracy on the COVID-AAAI dataset is
achieved using a COVID-twitter-BERT model,
which was trained on a large corpus of COVID-
related tweets, and outperforms our best model by
3.30% (Glazkova et al., 2020). As explained later,
with the cost of adding more in-domain data, the
performance tends to improve. Thus, we may have
achieved a better score if we would have trained on
more in-domain data.

For the ENRON email spam dataset, our ensemble
model performs the best amongst all the models.
It outperforms the SOTA hybrid network for spam
email detection by 3.70% in F1-Score (Douzi et al.,
2020). Our ensemble model achieves the best accuracy in the
SMS Spam dataset, which outperforms our best model by
0.78% (Almeida et al., 2011).

The soft decision ensemble model does not per-
form better on five of the ten datasets than the stand-
alone models. As the ensemble model takes an
average of the softmax decision of the models, a
weak classifier gets an equal weight to a strong
classifier, which in turn may hurt the final decision.
Further investigation may be undertaken to develop
a weighted average ensemble of the models for a
more robust classifier.

The superior performance of the BERT model
information by the fine-tuning and the pooling pro-
cess, which creates further research direction to-
towards fine-tuning the SBERT model for deception
detection. We find the best overall performance for
the ensemble model, with an accuracy of 93.42%,
and an F1-Score of 93.22%. The best performing
standalone model – BERT lags slightly behind that
with an accuracy of 92.72%, and an F1-Score of
92.50%.

For the PHEME dataset, we find the best performing F1-Score in the ensemble model as
82.74%, which is better than the current top scorer
staA-HitPLAN based model (77.4%) (Khoo et al.,
2020). Similarly, for the Liar dataset, the ensemble
model achieves the best performance with an accuracy of 70.72%, and F1-Score of 62.60%, which
outperforms a text-based BERT-CNN architecture
by 5.18% and 1.80% respectively (Upadhayay and
Behzadan, 2020).

The current SOTA F1-Score in FakeNewsNet–
Gossipcop (FNN-Gossipcop) and Politifact (FNN-
Politifact) is 75.50% and 92.80% in F1-Score using
a news and user-comment encoder, and co-attention
network (Shu et al., 2019). However, their experiments differ from ours in the fact that i) they used
the news with at least three user comments, reduc-
ing their sample size by 73% and 60% compared to
the original data size, and ii) besides the news text,
they took user comments into account as well. The
user comments contain crowd opinion, which pro-
vides vital information to detect deceptive content
(Guo et al., 2018; Shu et al., 2019). Nevertheless,
selecting the source with user comments can sig-
nificantly reduce the data space, and thus, we do
not use them.

Our SBERT model performs the best in the
Rashkin-Politifact dataset, which outperforms the
baseline model provided in the paper by 37.46% (Rashkin et al., 2017). Our model performs the best
in the Rashkin-Newsfiles dataset in discriminating
between Satire and Hoax-Propaganda, achieving
an accuracy of 99.43%, and an F1-Score of 99.71%
in the BERT model.

For the COVID datasets, the BERT model out-
performs the SOTA hybrid network for deception
detection by 3.70% in F1-Score (Douzi et al.,
2020). Our ensemble model achieves the best accuracy in the
SMS Spam dataset, which outperforms the
baseline model provided in the paper by 37.46% (Rashkin et al., 2017). Our model performs the best
in the Rashkin-Newsfiles dataset in discriminating
between Satire and Hoax-Propaganda, achieving
an accuracy of 99.43%, and an F1-Score of 99.71%
in the BERT model.
comes from the self-attention layer in the transformer blocks, which is also confirmed by Vashishth et al. (Vashishth et al., 2019). We take the [CLS] token output as the feature vector, and thus the attention heads in each layer for the [CLS] token may have an important impact. We randomly take the deceptive text “if you have bank account or you can open new one then we need you!” into account and visualize the attention weights of [CLS] token. Figure 2 depicts the average attention weights of all attention heads in the final layer. Due to the averaging effect, apart from the [SEP] token, all the words show a close attention weight. However, different attention head focuses on a different part of the text, e.g., we observe that last two attention heads focus on the words ‘have’, ‘account’, ‘open’, ‘need’, and ‘!’, while the third head focuses on the word ‘you’, and “bank”. With a complex mechanism of self-attention and feed-forward network, the BERT model represents the sentence as a 768-D vector, which is used for the downstream deception-detection task.

We analyze the misclassified samples by the best performing standalone model–BERT. In Figure 1, we plot the BERT embeddings using t-SNE. We observe that BERT does not perform well when deceptive and non-deceptive text has overlapping embedding, which indicates a limitation of the textual feature representation by the BERT model. Future research might be undertaken to develop a variant of the BERT model that can better distinguish between deceptive and non-deceptive text.

To further analyze the type of misclassified texts, we randomly select False Positive examples (Not deceptive but predicted to be deceptive).
the name “Trump” has a false positive rate (FPR) of 23.17%, “Obama” has an FPR of 16.18%. On the contrary, the non-political names like “Gates” have an FPR of 2.01%. These findings suggest that the models may suffer from bias towards political names.

Next, we analyze the True Negatives (deceptive, but all our models predicted it to be non-deceptive). We randomly select the following samples:

1. You have received your mobile content. Enjoy
2. Celebrities slam Trump decision to end DACA as ‘callous,’ ‘disgusting,’ and a ‘grave mistake’
3. I’ve been here almost every day.
4. Forty-five percent of doctors say they’ll quit if health care reform passes
5. Says 57 percent of federal spending goes to the military and just 1 percent goes to food and agriculture, including food stamps

From the True-Negative samples, we observe a wide variety of examples that were misclassified to be non-deceptive. For the first sample, the models probably expect more persuasion to detect the deception. The third sample is a statement by the Missouri governor which was a lie. We infer that it may be hard for any model to detect a text as deceptive without proper context. These findings raise intriguing questions regarding the extent of the deceptive text, and for the model to successfully detect deception, maybe the context should be a part of the text.

Therefore, based on the overall analysis, the results of the holistic deception detection task supports H1.

5.2 New-event Deception Detection

The holistic model with a complete set of out-of-domain data and a fraction of in-domain data can perform well enough to detect the deceptive text on a new event like the COVID-19 pandemic. From Figure 3, we observe that BERT, SBERT, and Char-CNN give F1-Score performance of 67.96%, 62.70%, and 52.39% respectively while having no knowledge on the COVID event, which indicates a cold-start (Adomavicius and Tuzhilin, 2005). However, when added with only 20% in-domain COVID training data, the performance improves sharply to 94.50%, 90.38%, and 87.69%, with an average improvement of 29.84%. From that point, we gradually increase the in-domain training data by 20%, and we find the optimal performance by adding 100% in-domain data. Nevertheless, our holistic model achieves 95.40% of optimal performance by only seeing the 20% in-domain training data. Thus, our results support H2.

6 Conclusion and Future Work

This research presents the holistic deception detection technique where we intend to find a domain-independent system to detect deception. Our general holistic approach outperforms some of the benchmark datasets for deception detection, where we observe the superior performance of the BERT model. Additionally, we analyze the strength of our holistic approach in case of a new event like COVID-19. We show that an out-of-domain general training set with a small fraction of the in-domain training set can help achieve satisfactory performance. Based on our work, there can be several directions for further research –

- The BERT and SBERT model work for 512 tokens only. For deception within the long text, models like Longformer, DocBERT can be used (Beltagy et al., 2020; Adhikari et al., 2019).
- It is not clear which part of the text may contain the cues to be a deceptive one. Thus, researchers can investigate to localize deception.
• The current pre-trained models are not free of bias, which we also observe in this work. Further research can be done to avoid the bias.

• The analysis of how deception occurs within the text is still not a clearly studied area. Thus, deceptive text generation can unravel many unexplored areas. Besides, we can investigate certain psycho-linguistic traits like fear, greed, persuasion within the text and quantify these attributes for a stronger holistic deception detection model.

Acknowledgments

Research was supported in part by grants NSF 1838147, NSF 1838145, ARO W911NF-20-1-0254. The views and conclusions contained in this document are those of the authors and not of the sponsors. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References


Anna Glazkova, Maksim Glazkov, and Timofey Trifonov. 2021. g2tmn at constraint@aaai2021: Exploiting ct-bert and ensembling learning for covid-19 fake news detection.


Ling Min Serena Khoo, Hai Leong Chieu, Zhong Qian, and Jing Jiang. 2020. Interpretable rumor detection in microblogs by attending to user interactions.


Qian Li, Qingyuan Hu, Youshui Lu, Yue Yang, and Jingxian Cheng. 2019. Multi-level word features based on cnn for fake news detection in cultural communication. Personal and Ubiquitous Computing, pages 1–14.


V. Metsis, Ion Androutsopoulos, and G. Paliouras. 2006. Spam filtering with naive bayes - which naive bayes? In CEAS.


Arkaitz Zubiaga, Geraldine Wong Sak Hoi, Maria Liakata, and Rob Procter. 2016. PHEME dataset of rumours and non-rumours.
Towards Domain-Generalizable Paraphrase Identification by Avoiding the Shortcut Learning

Xin Shen
The Chinese University of Hong Kong
xshen@se.cuhk.edu.hk

Wai Lam
The Chinese University of Hong Kong
wlam@se.cuhk.edu.hk

Abstract

In this paper, we investigate the Domain Generalization (DG) problem for supervised Paraphrase Identification (PI). We observe that the performance of existing PI models deteriorates dramatically when tested in an out-of-distribution (OOD) domain. We conjecture that it is caused by shortcut learning, i.e., these models tend to utilize the cue words that are unique for a particular dataset or domain. To alleviate this issue and enhance the DG ability, we propose a PI framework based on Optimal Transport (OT). Our method forces the network to learn the necessary features for all the words in the input, which alleviates the shortcut learning problem. Experimental results show that our method improves the DG ability for the PI models.

1 Introduction

Paraphrase Identification (PI) is the task of recognizing whether one text is a restatement of another text, preserving the same meaning while adopting a different expression (Bhagat and Hovy, 2013). Neural network based models have been proposed for the supervised PI task, and achieve decent performance in the single-domain setting (Yin and Schütze, 2015; Wang et al., 2017; Yang et al., 2019). At present, the existing PI corpora are restricted to several particular domains (Dolan et al., 2004; Xu et al., 2014; He et al., 2020), while the practical sentence pair for the paraphrase judgment can be from any unlabeled domain. At the same time, building a PI corpus for a novel domain needs massive human effort and is expensive. Therefore, a natural question is: for the supervised models trained in the domains with annotated PI corpora, to what extent can they generalize to an out-of-distribution (OOD) domain?

In this paper, we investigate the multi-source (Blanchard et al., 2011) Domain Generalization (DG) (Wång et al., 2021; Zhou et al., 2021) problem for supervised PI. More specifically, we try to learn a PI model based on information from several annotated source domains, and it could generalize well to an unlabeled domain. We investigate several competitive PI models in the DG setting, and observe that their performance deteriorates dramatically when tested in an OOD domain. We conjecture that the poor performance is caused by the models’ tendency to the shortcut learning (Geirhos et al., 2020). More specifically, these models are prone to relying on the shortcut features, e.g., some cue words, for classification. These shortcut features are often unique in one particular dataset or domain. When tested on an OOD domain, the models’ performance deteriorates because the shortcuts are missing. Interestingly, this phenomenon is also observed in other NLP tasks or models, such as NLI (Gururangan et al., 2018; Du et al., 2021), reading comprehension (Kaushik and Lipton, 2018; Lai et al., 2021), and BERT (Ni- ven and Kao, 2019).

The PI models usually follow the sentence-pair classification paradigm (Lan and Xu, 2018; Devlin et al., 2019). Some models originally proposed for the other sentence pair classification tasks, e.g., Semantic Textual Similarity (STS) or Natural Language Inference (NLI) can be easily adapted to PI. In what follows, we directly apply the suitable models without further clarification. One general characteristic of these models is: they have a component of information aggregation, i.e., the extracted and encoded features are aggregated into one fixed-length vector before computing the loss function. We point out that this step is one cause of the shortcut learning problem. Because in this step, it is not uncontrollable that which features should be preserved or discarded in the aggregated vector. Inspired by this, we conduct a new design for the final network layer of PI models to improve the DG
ability. The motivation of our method is: if we can force the PI model to learn the necessary features for all the input words instead of just relying on the domain-specific shortcuts, then the effects of shortcut learning can be alleviated. To this end, our proposed network layer outputs the importance scores and contextualized representations for all the input words, and adopt the Optimal Transport (OT) (Villani, 2008) distance to decide whether two sentences are paraphrase or not. The resulting PI models can be trained end-to-end, the feature extraction and encoding layers are not affected. In the experiments, the PI data from four different domains are adopted for simulating the DG setting. To validate the effectiveness of our method, we consider two representative PI models and equip them with our proposed module. The evaluation results show that our method improve the OOD domain generalizability of these PI models.

2 Problem Formulation

The PI corpus is usually organized as a set \{((x_i, \bar{x}_i), y_i)\}_{i=1}^N. For each tuple \((x, \bar{x}), y\), \(x\) and \(\bar{x}\) are two input sentences, the label \(y = 1\) indicates that \(x\) and \(\bar{x}\) are the paraphrase while \(y = 0\) denotes the non-paraphrase. The associated domain of this dataset is defined as a joint distribution \(P_{XY}\) on \(\mathcal{X} \times \mathcal{Y}\), where \(\mathcal{X}\) is the space of input sentence pairs\(^1\), and \(\mathcal{Y}\) is the label space. Then the target of a PI model is to learn a function \(f: \mathcal{X} \rightarrow \mathcal{Y}\), which predicts the label \(y\) based on the sentence pair \((x, \bar{x})\).

We adopt the common setting of multi-source DG as in Blanchard et al. (2011). Specifically, assume that we can access a set of \(K(K > 1)\) distinct source domains \(\mathcal{S} = \{S^k\}_{k=1}^K\). Each \(S^k\) is associated with a distinct joint distribution \(P_{\mathcal{X}Y}^k\), i.e., \(P_{\mathcal{X}Y}^k \neq P_{\mathcal{X}Y}^{k'}, \forall k \neq k'\) and \(k, k' \in \{1, \ldots, K\}\).

For \(S^k\), the associated dataset contains i.i.d. data \(\{(x_i^k, \bar{x}_i^k), y_i^k\}_{i=1}^{N_k}\) sampled from \(P_{\mathcal{X}Y}^k\). The target domain denoted as \(\mathcal{T}\) is associated with a joint distribution \(P_{\mathcal{X}Y}^T\), where \(P_{\mathcal{X}Y}^T \neq P_{\mathcal{X}Y}^k, \forall k \in \{1, \ldots, K\}\). Then DG problem for PI is defined as: given the labeled source domains \(\mathcal{S}\), we try to learn a model based on information from \(\mathcal{S}\) such that the model can generalize well to an unseen domain \(\mathcal{T}\). It should be noted that DG is more challenging than the related settings such as domain adaptation (Patel et al., 2015) or transfer learning (Pan and Yang, 2009). The difference primarily lies in that DG cannot access both the feature distribution and the label distribution of the target domain \(\mathcal{T}\), which makes it more practical for real-world applications.

3 Method Description

3.1 Shortcut Learning Problem

Regardless of the implementation differences, most neural models for the supervised learning of PI follow the sentence pair classification paradigm, i.e., the features from two sentences are extracted and encoded into one vector for the classification (Lan and Xu, 2018). For these approaches, the final output representation fuses features from all the input words together, and we conjecture that it is the reason for the poor OOD generalizability. Concretely, these models are prone to the shortcut learning, i.e., utilizing the features from some cue words that are specific to the training domains. In the fused representations of the final output, the model neglects the features from the words that actually decide whether two sentences are paraphrase or not, because the model already makes the correct decision based on the shortcut features. When these shortcuts are missing in the OOD domain, the model performs poorly.
s1: In only 14 days\(^1\), **US researchers** have created\(^2\) an artificial bacteria-eating virus\(^3\) from synthetic genes\(^4\).

s2: An artificial bacteria-eating virus\(^3\) has been **made**\(^2\) from synthetic genes\(^4\) in **the record time** of just two weeks\(^1\).

**Label**: paraphrase;  
**Domain**: news;  
**Dataset**: MRPC (Dolan et al., 2004).

s3: **how**\(^1\) the optimal solution\(^2\) to a linear programming problem\(^3\) changes\(^4\) as the **problem data** are modified.

s4: **how**\(^1\) changes\(^4\) in the **coefficients** of a linear programming problem\(^3\) affect the optimal solution\(^2\).

**Label**: non-paraphrase;  
**Domain**: computer science;  
**Dataset**: PARADE (He et al., 2020).

Table 1: Examples of paraphrase and non-paraphrase text pairs, which come from two different domains. We manually annotate the phrase-to-phrase alignment, and the semantically related phrases are annotated with the same superscript. For the sake of brevity, we do not annotate more detailed word-to-word alignment, and some unimportant words such as stop words are not annotated. We use red italic font to denote the words that cannot be suitably aligned.

### 3.2 OT Distance for Measuring the Text Similarity

As a preliminary to our method, we introduce the OT distance first. It provides an explainable approach to measuring the text similarity. Concretely, given two pieces of texts \(x = [w_1, w_2, \ldots, w_m]\) with \(m\) words and \(\bar{x} = [\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_n]\) with \(n\) words, the OT distance between \(x\) and \(\bar{x}\) is defined as:

\[
D_{OT}(x, \bar{x}) = \min_{P \in \Pi(\mu, \tilde{\mu})} \langle P, C \rangle = \min_{P \in \Pi(\mu, \tilde{\mu})} \sum_{i=1}^{m} \sum_{j=1}^{n} c(w_i, \bar{w}_j) \cdot P_{ij}. \tag{1}
\]

Here, \(C\) stands for the cost matrix, whose element \(C_{ij} = c(w_i, \bar{w}_j)\) determines the cost of transporting the word \(w_i\) to the word \(\bar{w}_j\). \(c(w_i, \bar{w}_j)\) is smaller when \(w_i\) and \(\bar{w}_j\) are more semantically similar. The matrix \(P\) is the transport plan, where \(P_{i,j}\) is larger when \(w_i\) and \(\bar{w}_j\) are more closely aligned. \(\Pi(\mu, \tilde{\mu})\) is the set of all the feasible transport plans. \(\langle \cdot, \cdot \rangle\) stands for the Frobenius dot product between two matrices of the same size. The vectors \(\mu = [\mu_1, \ldots, \mu_m]\) and \(\tilde{\mu} = [\tilde{\mu}_1, \ldots, \tilde{\mu}_n]\) satisfy that \(\sum_{i=1}^{m} \mu_i = \sum_{j=1}^{n} \tilde{\mu}_j = 1\), and the element \(\mu_i\) or \(\tilde{\mu}_j\) reflects the relative importance of the corresponding word in the text.

In Figure 1, we give an example of OT distance between two sentences. From this example, we can observe that: by solving the optimization problem in Formula (1), the solution matrix \(P\) explicitly aligns the semantically related words. And the optimal objective value of Problem (1) is the distance of moving sentence \(x\) to sentence \(\bar{x}\).

For the values of vectors \(\mu\) and \(\tilde{\mu}\), different models behave differently. Word Mover’s Distance (WMD) (Kusner et al., 2015) requires that all the words in one text are equally treated, i.e., \(\mu_i = \frac{1}{m} (\forall i, 1 \leq i \leq m)\), and \(\tilde{\mu}_j = \frac{1}{n} (\forall j, 1 \leq j \leq n)\). Yokoi et al. (2020) point out that WMD is not suitable for unsupervised STS, because the importance of each word should be differentiated. They propose Word Rotator’s Distance (WRD) and adopt the norm of the pretrained embedding as the weight of the corresponding word. For the unsupervised methods WMD and WRD, the values of \(\mu\) and \(\tilde{\mu}\) are fixed, and independent of whether the sentence pair \((x, \bar{x})\) is paraphrase or not. However, we claim that they depend on each other in the supervised setting. Consider some examples in Table 1. Since the nature of paraphrase is semantic equivalence, for the paraphrase sentences (e.g., s1 and s2 in Table 1), the unaligned words (e.g., **US researchers** and **the record time**) are unimportant. Conversely, the key to a non-paraphrase is to find the difference in between. For the non-paraphrase sentences (e.g., s3 and s4 in Table 1), the unaligned words (e.g., **problem data** and **coefficients**) are key to make the difference and are
Algorithm 1 Log-domain Sinkhorn algorithm for computing the entropy-regularized OT distance.

Input: $k = 0$, $u^0 = 0_m$, $v^0 = 0_n$, $K$ is the maximum number of iterations allowed.

1: while $k < K$ do
2: $u^{k+1} = u^k + \varepsilon \log(\mu) - \log (R(u^k, v^k)1_n)$. 
3: $v^{k+1} = v^k + \varepsilon \log(\tilde{\mu}) - \log (R(u^{k+1}, v^k)1_m)$. 
4: $k = k + 1$.
5: end while
Output: $P^* = R(u^k, v^k)$.

thus important. Another important issue is the value of $c(w_i, \tilde{w}_j)$. For the unsupervised methods such as WMD and WRD, the value of $c(w_i, \tilde{w}_j)$ is fixed. It is usually computed based on the pretrained embeddings of the corresponding words. However, this practice lacks flexibility when representing the word relatedness in the supervised setting. The contextualized word representations should be adopted.

3.3 Domain-Generalizable PI via OT layer

The analysis in Section 3.1 indicates that the shortcut learning problem is caused by the aggregated representation in the classifier layer, which is adopted by most existing PI models. To make the PI models more domain-generalizable, we change the output layer of the network to memorize the necessary features of all the words during the in-domain training. The analysis in Section 3.2 suggests that the values of $\mu$, $\tilde{\mu}$, and $C$ should be adaptive in the supervised setting of PI. At the same time, these values are all specific to individual word. Therefore, we parameterize the word importance vectors and the contextualized word embeddings as the learnable outputs of a neural network. The neural network is trained so that the OT distance $D_{OT}(x, \tilde{x})$ is minimized for the paraphrase, while is maximized for the non-paraphrase:

$$
\begin{align*}
\begin{cases}
\min \ D_{OT}(x, \tilde{x}) & \text{if } y = 1; \\
\max \ D_{OT}(x, \tilde{x}) & \text{if } y = 0.
\end{cases}
\end{align*}
$$

(2)

In this way, we force the network to memorize representations for each individual word, instead of learning a fused representation. And we expect the shortcut learning problem can be alleviated. For the practical usage, we adopt the following regression based objective:

$$
\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \left\{ (\exp(-D_{OT}(x_i, \tilde{x}_i)) - y_i)^2 \right\},
$$

(3)

where $\theta$ denotes the network parameters. The objective in Formula (3) is mathematically equivalent to the objective in Formula (2), but is more numerically stable. Following the practice as in Chen et al. (2019, 2020), we compute the value of cost $c(w_i, \tilde{w}_j)$ as the cosine distance between the corresponding contextualized word representations. We name our method as Domain Generalizable Optimal Transport (DG-OT) layer. Except the final output layer, the preceding layers of PI models can be unchanged. In Figure 2, we present the architecture of decomposable attention model (Parikh et al., 2016) when equipped with our proposed DG-OT layer.

3.4 Computing the OT Distance

To incorporate the OT distance into neural networks, we adopt the practice in (Cuturi, 2013; Frognier et al., 2015) and solve the following entropy-regularized OT problem:

$$
\min_{P \in \mathcal{P}([\mu, \tilde{\mu}])} \langle P, C \rangle - \varepsilon H(P).
$$

(4)

Here, $H(P)$ is the entropy regularization term defined as: $H(P) = -\sum_{i,j} P_{ij} \log(P_{ij}) - 1$. $\varepsilon$ is a positive hyper-parameter for controlling its relative importance. When the value of $\varepsilon$ is small enough, Problem (4) is a good approximation of original OT distance in Formula (1). In this paper, we utilize the Sinkhorn algorithm in the log domain (Chizat et al., 2018; Schmitzer, 2019) to solve Problem (4). The details are presented in Algorithm 1, in which the function $R(u, v)$ is defined as: $R(u, v) = diag(\exp(\frac{u}{\varepsilon})) \exp(\frac{-v}{\varepsilon}) diag(\exp(\frac{u}{\varepsilon}))$. After computing the entropy-regularized OT distance between $x_i$ and $\tilde{x}_i$ with Algorithm 1, we substitute $D_{OT}(x_i, \tilde{x}_i)$ in Formula (3) with the objective value of Problem (4). The resulting OT classifier layer is fully differentiable, and the whole PI model can be trained in an end-to-end way.
Compute OT distance
Softmax

Sentence pairs:

Figure 2: The architecture of decomposable attention model (Parikh et al., 2016) when equipped with our proposed DG-OT layer.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Domain</th>
<th>#Training total (+/-)</th>
<th>#Validation total (+/-)</th>
<th>#Testing total (+/-)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRPC</td>
<td>News</td>
<td>4076 (2753/1323)</td>
<td>500 (250/250)</td>
<td>600 (300/300)</td>
</tr>
<tr>
<td>PIT-2015</td>
<td>Twitter</td>
<td>5000 (2500/2500)</td>
<td>1000 (500/500)</td>
<td>1200 (600/600)</td>
</tr>
<tr>
<td>QQP</td>
<td>Quora questions</td>
<td>5000 (2500/2500)</td>
<td>1000 (500/500)</td>
<td>1200 (600/600)</td>
</tr>
<tr>
<td>PARADE</td>
<td>Computer science</td>
<td>5000 (2500/2500)</td>
<td>1000 (500/500)</td>
<td>1200 (600/600)</td>
</tr>
</tbody>
</table>

Table 2: Statistics of the processed PI datasets. The symbol + indicates the paraphrase sentence pairs, while the symbol - indicates the non-paraphrase sentence pairs.

4 Experiment

4.1 Datasets and Settings

We consider four publicly available PI datasets from different domains for the experiment:

- **Microsoft Research Paraphrase Corpus (MRPC)** (Dolan et al., 2004), which contains sentence pairs from news articles.
- **Paraphrase Identification from Twitter (PIT-2015)** (Xu et al., 2014), which contains pairs of Twitter tweets.
- **Quora Question Pairs (QQP)**, which contains Quora question pairs.
- **PARAphrase identification based on Domain knowledge (PARADE)** (He et al., 2020), which contains definitions of terminologies from the domain of computer science.

To simulate the DG setting, we use three datasets for the in-domain training, and use the remaining one dataset for evaluating the OOD generalization ability. During the in-domain training stage, the validation set is merged from three in-domain validation sets. We also conduct the in-domain testing for the purpose of comparison, where the in-domain testing set is merged from three in-domain testing sets. To prevent the PI models from being dominated by one or several particular domains, we process the datasets so that each domain has relatively the same number of sentence pairs. Because the original splittings of these four datasets differ, and it is hard to directly sample training/validation/testing sets and ensure they are of comparative and relatively-large size over different domains. Therefore, for each
dataset, we merged the original splittings of training/validation/testing sets together, and randomly sample the new training/validation/testing sets. We conduct sampling without replacement. The statistics of the processed datasets are described in Table 2. Following the previous works, we adopt accuracy and F1 score as the evaluation metric.

### 4.2 Baselines

We adopt the following models for the experiment:

- **DECATT** (Parikh et al., 2016), a decomposable attention model. We change the original three-way classification to two-way classification.

- **BiMPM** (Wang et al., 2017), a bilateral multi-perspective matching model.

For these models, we adopt and adapt the implementations in AllenNLP-Models\(^3\). To validate the effectiveness of DG-OT layer, we equip these two models with DG-OT layer, and compare them with their vanilla versions. To be fair, the shared network structures have the same size. **DECATT** should be compared with **DECATT+DG-OT**, and **BiMPM** should be compared with **BiMPM+DG-OT**, when the other settings are the same. For all the methods, we adopt GloVe (Global Vectors for Word Representation)\(^4\) to initialize the word embeddings. The hyper-parameters are tuned based on the performance in terms of F1 on the validation set.

### 4.3 Results

The results of in-domain testing and OOD generalization are reported in Table 3, from which we can draw the following conclusions:

- When other conditions are the same, the OOD performance is poorer than the in-domain performance, which agrees with the common expectation. The reason is the underlying data distributions are different.

- In the same setting, the performance of BiMPM is generally better than that of DECATT. It suggests that BiMPM is more suitable for the PI task in the setting of multi-domain training.

- When equipped with DG-OT, both the performance of DECATT and BiMPM are improved obviously in both the in-domain and OOD setting. And the performance dropping brought by OOD domain is less obvious when DG-OT is equipped by the model. These results validate that DG-OT helps to avoid the shortcut learning.

### 5 Conclusions and Future Works

As a preliminary attempt in this direction, we investigate the DG problem for supervised PI task in this paper. We point out that the aggregation operation is one reason for the poor OOD generalization ability of the existing PI models. We incorporate the Optimal Transport (OT) distance and design a novel classifier layer, i.e., DG-OT layer. It tackles the shortcut learning problem by enforce the network to learn the importance weights and contextualized representations for all the words. The experiments validate the effectiveness of DG-OT layer. Avoiding the shortcut learning is only one factor to the DG ability of PI models. Another important aspect is how to handle the domain shift, which is left as our future work. Besides, sentence pair classification include other tasks such as STS and NLI. It is still unclear whether our method is suitable for STS and NLI, and we leave this topic as the future work.

<table>
<thead>
<tr>
<th>Method</th>
<th>source: NTQ → NTQ</th>
<th>source: NTC → TNC</th>
<th>source: NQC → TQC</th>
<th>source: TQC → TQC</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECATT</td>
<td>56.7/62.9/50.4</td>
<td>65.5/62.4/56.4</td>
<td>67.1/63.2/52.9</td>
<td>57.3/63.2/58.2</td>
</tr>
<tr>
<td>DECATT+DG-OT</td>
<td>63.9/68.8/65.4</td>
<td>70.9/68.4/65.4</td>
<td>65.2/62.8/57.1</td>
<td>71.8/73.5/59.7</td>
</tr>
<tr>
<td>BiMPM</td>
<td>65.2/66.7/62.1</td>
<td>68.8/64.9/61.2</td>
<td>67.1/68.2/63.5</td>
<td>72.6/73.2/64.5</td>
</tr>
<tr>
<td>BiMPM+DG-OT</td>
<td>67.1/70.3/64.1</td>
<td>63.8/67.5/61.9</td>
<td>67.2/68.2/65.3</td>
<td>71.7/68.2/64.9</td>
</tr>
</tbody>
</table>

Table 3: Results of in-domain testing and OOD generalization. Each result is organized as **accuracy/F1**. We use the initials N, T, Q, C to represent the domains of News, Twitter, Quora, and Computer science, respectively.

\(^3\)https://github.com/allenai/allennlp-models.

Acknowledgments

The work described in this paper is substantially supported by a grant from the Asian Institute of Supply Chains and Logistics, the Chinese University of Hong Kong.

References


Czert – Czech BERT-like Model for Language Representation

Jakub Sido* and Ondřej Pražák* and Pavel Přibáň and Jan Pašek and Michal Seják and Miloslav Konopík

{sidoj,ondfa,pribanp,pasekj,sejakm,konopik}@kiv.zcu.cz

NTIS – New Technologies for the Information Society, Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, Technická 8, 306 14 Plzeň Czech Republic

Abstract

This paper describes the training process of the first Czech monolingual language representation models based on BERT and ALBERT architectures. We pre-train our models on more than 340K of sentences, which is 50 times more than multilingual models that include Czech data. We outperform the multilingual models on 9 out of 11 datasets. In addition, we establish the new state-of-the-art results on nine datasets. At the end, we discuss properties of monolingual and multilingual models based upon our results. We publish all the pre-trained and fine-tuned models freely for the research community.

1 Introduction

Transfer learning and pre-trained word embeddings became a crucial component for most Natural Language Processing (NLP) models. Contextualized methods (McCann et al., 2017; Peters et al., 2018; Howard and Ruder, 2018) overcame the initial context insensitive word embeddings approaches (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017). (McCann et al., 2017; Peters et al., 2018). The word representations generated by the named methods are usually used as input features for other task-specific models that are further trained. Starting with the BERT (Devlin et al., 2018), the BERT-like models (Lan et al., 2020; Liu et al.; Sanh et al., 2019; Yang et al., 2019) based on Transformer architecture (Vaswani et al., 2017), achieved a significant performance improvement in many NLP tasks (Raffel et al., 2019). These recent models are trained on a language model task or tasks that are closely related to it. Such pre-training allows them to capture the general representation of language and text. The pre-trained models are then directly fine-tuned with specific data for a selected downstream task. The performance improvement of these models is paid by the vastly increased requirements (i.e., data and computational resources) for their training.

The mentioned models are primarily trained for English. Recently, models for other, mostly larger, languages have been released, e.g., French (Martin et al., 2020; Le et al., 2019), Polish (Kleczyk, 2020), Turkish (Schweter, 2020), Russian, German, Arabic (Safaya et al., 2020), but also for languages that are spoken by a relatively small number of people, i.e., Romanian (Dumitrescu et al., 2020), Dutch (Vries et al., 2019) or Finish (Virtanen et al., 2019). There were also introduced multilingual models (Conneau and Lample, 2019; Conneau et al., 2020), that can be used for multiple languages at once but usually at the cost of lower performance in comparison to solely monolingual models (Martin et al., 2020; Virtanen et al., 2019; Dumitrescu et al., 2020) as we show in this paper.

Our main motivation is to train and provide publicly available models for the Czech language that performs better than available multilingual models.

In this paper, we describe a process of training of two BERT-like models for Czech language and their evaluation on six tasks along with a comparison to two multilingual models, i.e. mBERT (Devlin et al., 2018) and SlavicBERT (Arkhipov et al., 2019). More concretely, the architectures of our models are based on the ALBERT (Lan et al., 2020) model (Czert-A) and the original BERT (Devlin et al., 2018) model (Czert-B). Both of our models are trained on a text corpus of the approximate size of 36 GB of plain text consisting of Czech Wikipedia articles, crawled Czech news and Czech

1http://docs.deeppavlov.ai/en/master/features/models/bert.html
2https://github.com/dbmdz/berts
3The model is available at https://github.com/kiv-air/Czert
National Corpus (Křen et al., 2016). We train the models from scratch (i.e., with random initialization) using Masked Language Model (MLM) and Next Sentence Prediction (NSP) tasks as training objectives with a slight modification of the NSP task, see Section 3. We evaluate our models on six tasks: Semantic Text Similarity (STS), Named Entity Recognition (NER), Morphological Tagging (MoT), Semantic Role Labeling (SRL), Sentiment Classification (SC) and Multi-label Document Classification (MLC).

Our main contributions are the following ones: 1) We release a pre-trained and ready-to-use BERT model (Czert-B) for the Czech language that outperforms the compared models on all evaluated sentence-level tasks and it performs comparably on Semantic Role Labeling task. Along with the pre-trained model, we also release the fine-tuned models for each task. 2) We achieve new state-of-the-art results on seven datasets. Moreover we outperform the multilingual models with our newly trained Czert-B model on 7 out of 10 datasets.

2 Related Work

2.1 English BERT and ALBERT

The BERT (Devlin et al., 2018) model adopts the multi-layer Transformer-encoder architecture (Vaswani et al., 2017) with two pre-training tasks: Masked Language Modeling and Next Sentence Prediction.

The goal of the MLM task is to recover artificially distorted sentences where some of the original tokens are masked out (hidden), and some are randomly replaced with other tokens. These distorted tokens and few other unchanged tokens are selected for prediction (classification). The ratios of predicted tokens can be tuned. For example, in the original BERT model, 15% of input tokens are predicted, 80% of them are masked out, 10% are changed randomly, and 10% are left intact.

The NSP is a binary classification task of sentence pairs. For two sentences A and B taken from the training corpus, the goal is to decide whether the sentence B is the actual next sentence (following the sentence A) or whether it is a randomly selected sentence from the corpus. In the BERT paper (Devlin et al., 2018), the random sentences are sampled uniformly from the whole corpus.

The BERT model represents a big step in massively pre-trained models. The experiments\(^5\) show that a large stack of cross-attention layers with a huge amount of parameters of BERT and BERT-like models can significantly boost the performance of many downstream tasks. A relatively short fine-tuning phase is usually sufficient to set new state-of-the-art results in many tasks using the pre-trained model.

In the original paper (Devlin et al., 2018), the authors publish the BERT\textsubscript{BASE} and BERT\textsubscript{LARGE} models. BERT\textsubscript{BASE} contains 12 layers, 12 attention heads, and the size of the hidden state is set to 768. In total, it requires 110M parameters. The BERT\textsubscript{LARGE} model has 24 layers, 16 attention heads and the size of the hidden state is set to 1024, which results in 340M parameters.

Training such huge models requires vast computational resources. Therefore, researchers developed methods to reduce the training complexity, memory demands or prediction time, while maintaining similar performance on the fine-tuned tasks. ALBERT model (Lan et al., 2020) represents an example of such an approach.

ALBERT slightly modifies BERT to use the parameters more effectively. First, the authors argue that word embedding size equal to the hidden size (768 for base) is unnecessarily large. They propose to use a smaller size (128) and project the embeddings to the hidden size, which significantly reduces the number of parameters (25M less than in the base variant). Another modification is in cross-layer parameter sharing. In ALBERT, all the weights are shared across all the layers. Another modification consists of replacing the NSP task with a harder task of sentence ordering prediction (SOP). That should result in making the model understand semantics better. The authors introduce models ALBERT\textsubscript{BASE}, ALBERT\textsubscript{LARGE}, ALBERT\textsubscript{XLARGE}, ALBERT\textsubscript{XXLARGE} with 12M, 18M, 60M and 235M parameters, see Table 1.

2.2 BERT-like Models for Other Languages

Researchers publish a multilingual variant of standard BERT\textsubscript{BASE} model (mBERT)\(^6\). It is jointly trained on Wikipedia pages of 104 languages. The model settings are almost the same as in

\(^{5}\)Experiments in the BERT paper (Devlin et al., 2018) or in many consequent research papers.

\(^{6}\)See https://github.com/google-research/bert/blob/master/multilingual.md.
BERT\textsubscript{BASE}; it differs only in the vocabulary size\textsuperscript{7}.

However, researchers around the world trained the monolingual variant of the BERT and showed the domination of the monolingual version over the mBERT in many tasks, for example, French (Martin et al., 2020), Finish (Virtanen et al., 2019) or Romanian (Dumitrescu et al., 2020).

Arkhipov et al. (2019) used a combination of four Slavic languages: Bulgarian, Czech, Polish, and Russian. They trained their model using Wikipedia dumps for all four languages and a huge set of Russian news texts. They use the same model architecture and training process as mBERT, and they initialized the model with mBERT weights. Table 1 shows the sizes of each corpus in terms of quantities of raw text data, pre-process them and prepare automatically labeled training data.

Training corpora We use two publicly available corpora and our crawled dataset of Czech news:

- Czech national corpus (CsNat) 28.2GB, (Křen et al., 2016),
- Czech Wikipedia (CsWiki) 0.9GB, dump\textsuperscript{8} from May 2020,
- Crawled of Czech news (CsNews), 7.8GB.

The CsNat corpus composes of randomly-ordered blocks of texts sized maximum size of 100 tokens. Each block contains at least one sentence. This must be considered later for the NSP task, which requires a continuous block of texts. Table 3.1 shows the sizes of each corpus in terms of blocks and sentences counts.

Pre-processing We prepare two versions of the corpus: \textit{cased} and \textit{uncased}. Both versions are tokenized with the WordPiece tokenizer (Wu et al., 2016) which is trained on the entire corpus.

Pre-training Objective We employ MLM and NSP tasks (see section 2.1) for training our model.

The MLM task is used exactly as in the BERT model. The NSP task needs a few considerations. The NSP task requires the availability of continuous blocks of text to form pairs of sentences where one sentence follows the other. At the end of each block, we lose the last sentence that has no sentence to form a pair with. The effect of this issue becomes more apparent with the decreasing length of the continuous text blocks, such as in the case of the CsNat corpus. Here, we observe 5.6 sentences per continuous block on average. That means that we are able to use 4.6 sentences out of 5.6 (i.e. approximately 18% of sentences cannot form a pair). When compared to the two remaining corpora, this number is relatively high. In the CsWiki and CsNews corpora, only 6% and 4%, respectively, of sentences cannot form a pair.

Moreover, we design more difficult negative samples for the NSP task – we select sentences from the same paragraph (that do not directly follow the first sentence) to build non-trivial negative pairs instead of drawing random sentences from the whole corpora as in BERT.

The final dataset consists of 578 158 196 training pairs of sentences. In Table 3.1, we provide some basic statistics of the dataset used in our setup.

3  Pre-training Process
3.1 Dataset Description

Training BERT-like models require to collect large quantities of raw text data, pre-process them and prepare automatically labeled training data.

Training corpora We use two publicly available corpora and our crawled dataset of Czech news:

- Czech national corpus (CsNat) 28.2GB, (Křen et al., 2016),
- Czech Wikipedia (CsWiki) 0.9GB, dump\textsuperscript{8} from May 2020,
- Crawled of Czech news (CsNews), 7.8GB.

The CsNat corpus composes of randomly-ordered blocks of texts sized maximum size of 100 tokens. Each block contains at least one sentence. This must be considered later for the NSP task, which requires a continuous block of texts. Table 3.1 shows the sizes of each corpus in terms of blocks and sentences counts.

Pre-processing We prepare two versions of the corpus: \textit{cased} and \textit{uncased}. Both versions are tokenized with the WordPiece tokenizer (Wu et al., 2016) which is trained on the entire corpus.

\textsuperscript{7}BERT\textsubscript{BASE} uses a vocabulary with 30K sub-word tokens while mBERT increases the size to 120K tokens.

\textsuperscript{8}Taken from https://dumps.wikimedia.org

<table>
<thead>
<tr>
<th>Textual Blocks</th>
<th>Sentences</th>
<th>Avg/block</th>
</tr>
</thead>
<tbody>
<tr>
<td>CsNat</td>
<td>49 104 507</td>
<td>275 314 224</td>
</tr>
<tr>
<td>CsWiki</td>
<td>450 000</td>
<td>6 964 794</td>
</tr>
<tr>
<td>CsNews</td>
<td>2 625 306</td>
<td>58 979 893</td>
</tr>
</tbody>
</table>

Table 2: Statistics of corpora used.

3.2 Models

We train two models: a smaller ALBERT\textsubscript{BASE} model (Czert-A, 12M parameters) and a larger BERT\textsubscript{BASE} model (Czert-B 110M parameters).

Czert-A is very similar to the standard ALBERT\textsubscript{BASE} with a few modifications: we use WordPiece tokenizer, the batch size is set to 2048 (due to cluster limits), and we use our version of NSP introduced in Section 3 instead of SOP.

Czert-B is configured exactly as the BERT\textsubscript{BASE} model with increased batch size to 2048.

Optimization Both models are trained using a learning rate of 1e-4 with the linear decay using...
Adam optimizer (Kingma and Ba, 2014). First, we iterate over the dataset once (single epoch) with the maximum sequence length set to 128. It leads to 300K batches (steps). Similarly to the BERT approach, we then increase the maximum sequence length to 512. We perform about 50K steps with the increased sequence length. In this second shorter iteration, we decrease the batch to 256 samples to fit the cluster memory limits. More details about the computational cluster and its configuration are located in Appendix A.

4 Evaluation

The following section summarizes the performance of Czert on various tasks and compares our model with similar available models. We also add experiments without the pre-training phase to highlight the impact of additional unsupervised data in the Czech language. We also compare Czert with the following baselines:

**Baselines**

- SlavicBERT – a model trained on four Slavic languages (Russian, Bulgarian, Czech and Polish) (Arkhipov et al., 2019),
- mBERT – a multilingual version of BERT (Devlin et al., 2018),
- ALBERT-r – a randomly initialized ALBERT model without any pre-training.

4.1 Evaluation Tasks

We evaluate our models on six tasks that cover three main groups of NLP tasks: Sequence Classification (Sentiment Classification, Multi-label Document Classification); Sequence Pair Classification (Semantic Text Similarity); Token Classification (Morphological Tagging, Named Entity Recognition, Semantic Role Labeling)

For the **sequence classification** tasks, we take the pooled output of the BERT model (and ALBERT). We add dropout and an output layer. The number of output neurons and the activation function differs for each task.

**Sentence pair classifications** tasks employ the same approach as sequence classification tasks. The only difference is that we feed both sentences separated with special [SEP] token together into the model. This way, the model can profit from cross-attention between tokens from different sentences.

For the **token classification** tasks, we use the output embeddings associated with the input words ([CLS], [SEP] and other special output embeddings are ignored). When the input words are split to sub-word tokens, we take only the first sub-word tokens. For optimization, we use the Cross-entropy loss.

For all the tasks, the newly added layers are initialized randomly. We employ the Adam optimizer.

4.2 Named Entity Recognition

We use two different datasets to evaluate our model on the named entity recognition task. These are the following:

1. Czech Named Entity Corpus (CNEC) (Ševčíková et al., 2007) containing 4 688 training, 577 development and 585 test sentences. We use the CoNLL version of the dataset (Konkol and Konopík, 2013).
2. BSNLP 2019 shared task dataset (Piskorski et al., 2019) that consists of 196 train and 302 test sentences. We further split the test dataset into development and test parts resulting in development and test datasets of sizes 149 and 153 sentences, respectively. Additionally, we convert the original dataset into the same format as the CNEC, extracting entity classes only.

Independently on the dataset, we pre-process the sentences so that the maximum length of an example is 128 sub-word tokens. If the maximum length is exceeded, the residual part is used to create another data point. On the contrary, if the maximum length is not reached, the sentence is padded (padding is inserted at the end of the sentence). It is worth mentioning that exceeding the maximum length of a sentence occurs only for 44 times on the CNEC, which is negligible. On the other hand, on the BSNLP 2019, the length of the sentences differs a lot, and the maximum length is exceeded for a significant portion of the data. However, our experiments show that increasing the maximum sequence length does not improve the resulting F1 score. The architecture of the model follows the token classification settings described in Section 4.1. See Appendix B.1 for more details about the model and hyper-parameters.

4.2.1 Results

As an evaluation metric, we use F1 score computed on the entity level, while ignoring "O" (empty)
class. The results, stated with 95% confidence intervals, are summarized in Table 3.

<table>
<thead>
<tr>
<th>Class</th>
<th>CNEC</th>
<th>BSNLP 2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>mBERT</td>
<td>86.23 ± 0.21</td>
<td>84.01 ± 1.25</td>
</tr>
<tr>
<td>SlavicBERT</td>
<td>86.57 ± 0.12</td>
<td>86.70 ± 0.37</td>
</tr>
<tr>
<td>ALBERT-r</td>
<td>34.64 ± 0.34</td>
<td>19.77 ± 0.94</td>
</tr>
<tr>
<td>Czert-A</td>
<td>72.95 ± 0.23</td>
<td>48.86 ± 0.61</td>
</tr>
<tr>
<td>Czert-B</td>
<td>86.27 ± 0.12</td>
<td>86.73 ± 0.34</td>
</tr>
<tr>
<td>SoTA</td>
<td>81.77 b</td>
<td>93.9 4</td>
</tr>
</tbody>
</table>

Table 3: Comparison of F1 score achieved using pre-trained Czert-A, Czert-B, mBERT, SlavicBERT and randomly initialised ALBERT on NER task.

4.3 Morphological Tagging

To evaluate our model on a morphological tagging task, we utilize four Universal Dependencies treebanks. These are namely: Prague Dependency Treebank 3.0 (PDT) (Bejček et al., 2013), Czech Academic Corpus 2.0 (Vildová et al., 2008), Czech Legal Text Treebank 2.0 (Krň et al., 2018) and FicTree (Hnátková et al., 2017). Together they comprise 103 143 train, 11 326 development and 12 216 test examples. Furthermore, we also perform our experiments on the PDT only to compare our model to the current SoTA. The PDT dataset then comprises 68 627 train, 9 285 dev and 10 163 test examples. The original datasets come as CoNLL files which we converted to a simplified format as in the case of the CNEC dataset (section 4.2). During this pre-processing step, we extracted only UPOS tags, which we use as labels. The architecture of the model follows the token classification settings described in Section 4.1. The number of output neurons is set to the number of possible UPOS tags. See B.2, for more details about the hyper-parameters and training process.

4.3.1 Results

Table 4 shows the achieved results with 95% confidence intervals. Results are stated in F1 score computed on a token level, ignoring the "O" (empty) class. As the table shows, our model Czert-B outperforms the other models on both datasets. Moreover, we outperformed the current SoTA (Straka et al., 2019) as well.

<table>
<thead>
<tr>
<th>Universal Dependencies</th>
<th>PDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>mBERT</td>
<td>99.176 ± 0.006</td>
</tr>
<tr>
<td>SlavicBERT</td>
<td>99.211 ± 0.008</td>
</tr>
<tr>
<td>ALBERT-r</td>
<td>96.590 ± 0.096</td>
</tr>
<tr>
<td>Czert-A</td>
<td>98.713 ± 0.008</td>
</tr>
<tr>
<td>Czert-B</td>
<td>99.300 ± 0.009</td>
</tr>
<tr>
<td>SoTA</td>
<td>99.34 4</td>
</tr>
</tbody>
</table>

Table 4: Comparison of F1 score achieved using pre-trained Czert-A, Czert-B, mBERT, SlavicBERT and randomly initialised ALBERT on morphological tagging task. *Result is taken from (Straka et al., 2019).

4.4 Semantic Role Labelling

In semantic role labeling we are looking for shallow semantic structure so the task can be formalized as classification of roles arguments of the predicates in the sentence. Therefore, a single example to be classified is the pair of predicate and argument where the predicate is a single word, and the argument is either word or a phrase. We are classifying the role of the argument towards the predicate. Our input representation is inspired by (Shi and Lin, 2019). We first tokenize the sentence with WordPiece. Then we feed the sentence into the network followed by the [CLS] token and the predicate token(s). Note that the predicate tokens have the same positional IDs as their occurrence in the sentence, but different segment ids. This way the predicate at the end of the sequence differs from its in-sentence representation only in segment embedding, so it contains all the information to encode the in-sentence context but it can be easily distinguished from other tokens by the segment embedding.

4.4.1 Results

We evaluate Semantic role labeling for the Czech language on the CoNLL 2009 dataset. The results are shown in Table 5; the dep-based column denotes the result achieved by Zhao et al. (2009). In gold-dep, we replicated their system but evaluated it with gold-standard dependency trees. Syntax-based F1 metric is computed on whole subtrees of dependency trees. To compute this for span based model, we need to project labels on dependency trees. We did not optimize this projection in any way. We just removed B- and I- prefixes, we copied the dependency annotation and ran the
Table 5: SRL results – dep columns are evaluate with labelled F1 from CoNLL 2009 evaluation script, other columns are evaluated with span F1 score same as it was used for NER evaluation.

<table>
<thead>
<tr>
<th>Model</th>
<th>SPAN</th>
<th>SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>mBERT</td>
<td>78.55 ± 0.11</td>
<td>90.23 ± 0.22</td>
</tr>
<tr>
<td>SlavicBERT</td>
<td>79.33 ± 0.08</td>
<td>90.49 ± 0.04</td>
</tr>
<tr>
<td>ALBERT-r</td>
<td>51.37 ± 0.42</td>
<td>80.75 ± 0.13</td>
</tr>
<tr>
<td>Czert-A</td>
<td>76.63 ± 0.13</td>
<td>89.94 ± 0.05</td>
</tr>
<tr>
<td>Czert-B</td>
<td>81.86 ± 0.10</td>
<td>91.46 ± 0.06</td>
</tr>
<tr>
<td>dep-based</td>
<td>-</td>
<td>85.19</td>
</tr>
<tr>
<td>gold-dep</td>
<td>-</td>
<td>89.52</td>
</tr>
</tbody>
</table>

Table 6: Average F1 results for the Sentiment Classification task. The numbers in the brackets denote the initial learning rate and number of epochs, respectively, for training of the corresponding model. The state-of-the-art results are taken from (Habernal et al., 2013) and (Sido and Konopík, 2019).

<table>
<thead>
<tr>
<th>Model</th>
<th>FB</th>
<th>CSFD</th>
</tr>
</thead>
<tbody>
<tr>
<td>mBERT</td>
<td>71.72 ± 0.91 (2e-5 / 6)</td>
<td>82.80 ± 0.14 (2e-6 / 13)</td>
</tr>
<tr>
<td>SlavicBERT</td>
<td>73.87 ± 0.50 (2e-5 / 3)</td>
<td>82.51 ± 0.14 (2e-6 / 12)</td>
</tr>
<tr>
<td>ALBERT-r</td>
<td>59.50 ± 0.47 (2e-6 / 14)</td>
<td>75.40 ± 0.18 (2e-6 / 13)</td>
</tr>
<tr>
<td>Czert-A</td>
<td>72.47 ± 0.72 (2e-5 / 8)</td>
<td>79.58 ± 0.46 (2e-6 / 8)</td>
</tr>
<tr>
<td>Czert-B</td>
<td>76.55 ± 0.14 (2e-6 / 12)</td>
<td>84.79 ± 0.26 (2e-5 / 12)</td>
</tr>
<tr>
<td>SoTA</td>
<td>69.40</td>
<td>80.5 ± 0.165</td>
</tr>
</tbody>
</table>

4.5 Sentiment Classification

Sentiment Classification (SC) task (Liu, 2012) is a classification task where the goal is to assign a sentiment polarity of a given text. The positive, negative and neutral classes are usually used as the sentiment polarity labels. We perform the evaluation on two Czech sentiment classification datasets from Habernal et al. (2013), consisting of (1) Facebook posts and (2) movie reviews.

The Facebook dataset (FB) contains 10K users’ posts taken from nine Czech Facebook pages annotated with three classes.

We split the datasets into train, development and test parts with class distribution that follows the original datasets.

We fine-tune the models separately for each dataset. The architecture of the model follows the sequence pair classification setting described in Section 4.1. The number of output neurons is set to the number of sentiment polarity classes. Softmax normalization is applied to the output layer. We employ Cross-entropy loss. See B.4, for more details about hyper-parameters.

4.5.1 Results

We fine-tune the models (including the baselines) to achieve the best F1 score on the development data. Then, we use the best model settings to train a model on the train and development data. This model is evaluated on the test data and results are reported in Table 6 along with the initial learning rate and the number of epochs used for training. We repeat each experiment six times, and we report the average F1 score along with the 95% confidence interval.

4.6 Multi-label Document Classification

Multi-label Document Classification is a variant of classification problem where multiple labels can be assigned to each document. In this problem, there is no constraint on how many of the labels can be assigned to a given document.

We work with the Czech Text Document Corpus v 1.0 (Hrala and Král, 2013) to fine-tune and evaluate the models. The Czech News Agency provided almost 12 thousands of documents that formed the basis of this dataset. The agency journalists assign 60 categories (tags) to the documents as a part of their daily work. Following the approach from (Lenc and Král, 2018), we use only 37 most frequent categories for evaluation. More statistics are available in the paper.
<table>
<thead>
<tr>
<th>CTDC-1</th>
<th>AUROC</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>mBERT</td>
<td>97.62 ± 0.08</td>
<td>83.04 ± 0.16</td>
</tr>
<tr>
<td>SlavicBERT</td>
<td>97.80 ± 0.06</td>
<td>84.08 ± 0.14</td>
</tr>
<tr>
<td>ALBERT-r</td>
<td>94.35 ± 0.13</td>
<td>72.27 ± 0.17</td>
</tr>
<tr>
<td>Czert-A</td>
<td>97.49 ± 0.07</td>
<td>82.27 ± 0.17</td>
</tr>
<tr>
<td>Czert-B</td>
<td>98.00 ± 0.04</td>
<td>85.06 ± 0.11</td>
</tr>
</tbody>
</table>

Table 7: Results for Multi-label Document Classification on Czech Text Document Corpus v 1.0 dataset – AUROC and F1 measures. SoTA taken from (Lenc and Král, 2018).

### 4.6.1 Model Description and Fine-tuning

For **multi-label classification of documents** (MLC), we follow the sequence classification setting described in Section 4.1. The output layer is activated by the **sigmoid** function. The loss is the **Binary Cross-entropy** function. In the context of this task, documents are regarded as sentences trimmed to the maximum sequence length in tokens set to 512. We chose to pick the first N tokens in each document as our trimming strategy.

We run twenty 10-epoch-long training phases for each model and average the results. See B.6 for more details.

We use both standard **F1** and the **AUROC** (Melo, 2013) evaluation metrics. AUROC represents the overall ability of MLC models to distinguish between different classes without being biased by any constant threshold value. We use 95% confidence interval. We present the results in Table 7.

### 4.7 Semantic Text Similarity

We evaluate our model on semantic text similarity task on two different datasets.

1. **STS-SVOB** (Svoboda and Brychčín, 2018) contains two datasets: images descriptions (550 train and 300 test samples); and headlines (375 train and 200 test samples). We use the raw variant without any lemmatization or stemming.

2. **STS-CNA** was created during our experiments with this new model in cooperation with Czech News Agency and Charles University. STS-CNA contains s 138,556 hand-annotated sentence pairs (Sido et al., 2021).

### 4.7.1 Model Description and Fine-tuning

The architecture of the model follows the sequence pair classification setting described in Section 4.1. The number of output neurons is set to 1, and no activation function is applied to the output layer. We tried to keep hyper-parameters as close as possible between all experiments; however, we were forced to change them slightly in case of Czert-A and ALBERT-r. Also, the datasets have different nature; thus, we use different sets of hyper-parameters for each dataset. See B.5

We run ten experiments for each configuration and use 95% confidence interval. The tables Table 8 and Table 9 summarize the results. Table 8 shows that Czert-B model significantly outperforms the SoTA on SVOB-IMG dataset. In the SVOB-HL dataset, the models perform in par. We believe that the draw can be caused by reaching the annotation accuracy limit of this dataset.

We also observe a more stable and robust training on extremely small datasets; both Czert models are less prone to over-fitting than other tested models.

### 5 Discussion

We summarize the overall results of all evaluated tasks in Table 10. The first three columns contain the token classification tasks, the next two columns show results for sequence classification tasks, and the last column belongs to sequence pair classification task. We can observe that Czert-B model
We thoroughly evaluate our models on six common tasks: sentiment classification (SC) and semantic text similarity (STS). The increase is of ∼5% and ∼3% in both sentiment datasets and of ∼5% in one of the semantic similarity datasets. We also overcame SoTA in MLC.

6 Conclusion

In this work, we present two monolingual BERT-like models (BERT and ALBERT) for the Czech language. We train the models with the original MLM task and with a slightly modified NSP task. We thoroughly evaluate our models on six common tasks, and we compare them with other multilingual models. We include task-specific state-of-the-art models in our comparison. We outperform multilingual models with our newly trained Czert-B model on 9 out of 11 datasets. In addition, we establish the new state-of-the-art results on 9 datasets. The results show the strong performance of the Czert-B model on STS, MLC, SC, SRL, and MoT tasks.

As our paper confirms and as is shown in similar works, monolingual Transformer-based language models often overcome the multilingual ones.

Our models are publicly available for research purposes at our website and in the hugging face repository.

Acknowledgement

This work has been partly supported by ERDF "Research and Development of Intelligent Components of Advanced Technologies for the Pilsen Metropolitan Area (InteCom)" (no.: CZ.02.1.01/0.0/0.0/17 048/0007267); and by Grant No. SGS-2019-018 Processing of heterogeneous data and its specialized applications. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

References


Table 10: Summary of our results. The bold results denote the current SoTA results. The underlined results are the best result achieved directly by fine-tuning the BERT-like models. Values with the † symbol are the new SoTA results that we established in this paper. Results are taken from original papers.

---

<table>
<thead>
<tr>
<th>Task</th>
<th>Czert-B</th>
<th>Czert-A</th>
<th>SlavicBERT</th>
<th>mBERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NER (CNL)</td>
<td>96.59</td>
<td>87.58</td>
<td>88.79</td>
<td>90.23</td>
</tr>
<tr>
<td>MoT (CNEC)</td>
<td>91.89</td>
<td>87.50</td>
<td>86.27</td>
<td>86.27</td>
</tr>
<tr>
<td>Sentiment</td>
<td>85.06</td>
<td>84.79</td>
<td>84.74</td>
<td>84.79</td>
</tr>
<tr>
<td>Multiclass</td>
<td>83.74†</td>
<td>79.89†</td>
<td>79.99</td>
<td>79.99</td>
</tr>
</tbody>
</table>

---

The results in Table 10 with the † symbol.
Florence, Italy. Association for Computational Linguistics.


Michal Křen, Václav Cvrček, Tomáš Čapka, Anna Čermáková, Milena Hnátková, Lucie Chlumská, Tomáš Jelínek, Dominika Kováříková, Vladimír Petkevič, Pavel Procházka, Hana Skoumalová, Michal Škrabal, Petr Truneček, Pavel Vondřicka, and Adrian Zasina. 2016. SYN v4: large corpus of written czech. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.


Area under the ROC Curve

Bryan McCann, James Bradbury, Caiming Xiong, and Jeffrey Pennington, Richard Socher, and Christopher


A Cluster Configuration

We use distributed training to set the weights of Czert. For distributed pre-training we rely on the Czech national cluster Metacentrum\(^1\). We employ 16 machines, each with two NVIDIA TESLA T4 graphic cards, which results in 32 T4s in total.

For the Czert-A model, we use standard Tensorflow (Abadi et al., 2015) distributed training, which is based upon the gRPC standard. It takes 12 days to training Czert-A with this setting.

The Czert-B model contains almost ten times as many trainable parameters as the Czert-A model. It proved impractical to train Czert-A with the tools provided by Tensorflow alone. We employ the MPI messaging standard that communicates over the OmniPath network with a speed of 100Gb/s. The Horovod (Sergeev and Balso, 2018) library handles all the synchronization transfers of our distributed training. We are able to reach the speeds of 2400ms per batch with this setting, which is approximately five times faster than with standard gRPC via TCP/IP. We are able to train the Czert-B model in 8 days.

B Fine-tuning and Hyper-parameters

B.1 Named Entity Recognition

In all of our experiments, we use Adam optimizer with a learning rate of 5e-5 and a linear decay to zero. Additionally, the Czert-B model uses a learning rate warm-up during the first epoch. Similarly to our NER experiments (Section 4.2s), we use a maximum sequence length of 128 sub-word tokens. The models are trained with batch size 64 for 13 epochs. For Czert-A it takes about 8 hours and 15 minutes on an NVIDIA Tesla-T4 GPU.

B.3 Semantic Role Labeling

For fine-tuning, we use Adam optimizer with a learning rate of 5e-5 and a linear decay to zero. We use a maximum sequence length of 128 sub-word tokens. We train the model on 2 Tesla T4 graphic cards with batch size of 64 for 12 epochs.

B.4 Sentiment Classification

We perform fine-tune training of the models by minimizing the Cross-Entropy loss function using the Adam (Kingma and Ba, 2014) optimization algorithm with default parameters (\(\beta_1 = 0.9, \beta_2 = 0.999\)) and with a linear learning rate decay (without warm-up). We try three different initial learning rates, i.e., 2e-6, 2e-5 and 2.5e-5 for at most 14 epochs. We use a max sequence length of 64, batch size of 32 for the FB\(^1\) dataset and a max sequence length of 512 and batch size of 14 for the CSFD dataset.

B.5 Semantic Textual Similarity

For the CNA dataset, we train two epochs using a batch of size 50, and LR 1e-5 with linear decay to zero for each model except Czert-A for which we used 5e-6 for four epochs, which lead to slightly better results.

For smaller datasets (SVOB-img and SVOB-hl) we used LR 5e-6 and train on 14k batches.

For each experiment, we used Adam optimizer, L2 weight normalization, and learning rate warm-up during the first 500 batches.

B.6 Multi-label Document Classification

For each experiment, we first run a linear grid search through learning rate parameter L = {2e-5, 4e-5, ..., 10e-4} and a decision \(D = \{true, false\}\) whether to use a linear learning rate decay\(^1\) or to

---

\(^{14}\)See https://wiki.metacentrum.cz/wiki/Usage_rules/Acknowledgement

\(^{16}\)Arriving at 0 at the end of the last epoch.
keep the maximum learning rate constant until the last step. The learning rate achieved maximum after 500 steps of the warm-up phase. After the grid search was complete, we’ve run twenty 10-epoch-long training phases for each of the extended models and average the results.
Exploring German Multi-Level Text Simplification

Nicolas Spring, Annette Rios, Sarah Ebling
Department of Computational Linguistics, University of Zurich, Switzerland
{spring,arios,ebling}@cl.uzh.ch

Abstract

We report on experiments in automatic text simplification (ATS) for German with multiple simplification levels along the Common European Framework of Reference for Languages (CEFR), simplifying standard German into levels A1, A2 and B1. For that purpose, we investigate the use of source labels and pretraining on standard German, allowing us to simplify standard language to a specific CEFR level. We show that these approaches are especially effective in low-resource scenarios, where we are able to outperform a standard transformer baseline. Moreover, we introduce copy labels, which we show can help the model make a distinction between sentences that require further modifications and sentences that can be copied as-is.

1 Introduction

Simplified language is a variety of standard language characterized by reduced lexical and syntactic complexity, the addition of explanations for difficult concepts, and clearly structured layout.1 Among the target groups of simplified language are people with cognitive impairment and autism spectrum disorder, prelingually deaf and functionally illiterate people, and sometimes also foreign language learners and children (Bredel and Maaß, 2016).

Automatic text simplification (ATS), the process of automatically producing a simplified version of a standard-language text, was initiated in the late 1990s (Carroll et al., 1998; Chandrasekar et al., 1996) and has since then been approached by means of rule-based and statistical approaches. As part of the rule-based paradigm, the operations carried out typically include replacing complex lexical and syntactic units by simpler ones. The statistical paradigm so far has mainly conceptualized the simplification task as a case of monolingual (sentence-based) machine translation (MT), i.e., as one of converting standard-language into simplified-language sentences using MT techniques (Specia, 2010). However, while in bilingual parallel texts used for MT, the relation between source and target sentences is mostly 1:1, ATS usually requires n:m alignments with unaligned parts in-between.

ATS research has been documented for English (Zhu et al., 2010), Spanish (Saggion et al., 2015), Portuguese (Aluisio and Gasperin, 2010), French (Brouwers et al., 2014), Italian (Barlacchi and Tonelli, 2013), and other languages. Research on German is still sparse but has gained momentum in recent years due to a number of legal and political developments in German-speaking countries, such as the introduction of a set of regulations for accessible information technology (Barrierefreie-Informationstechnik-Verordnung, BITV 2.0) in Germany, the approval of rules for accessible information and communication (Barrierefreie Information und Kommunikation, BIK) in Austria, and the ratification of the United Nations Convention on the Rights of Persons with Disabilities (UN CRPD) in Switzerland.

In this paper, we report on work in automatic simplification of standard German into three separate simplification levels (A1, A2, B1) using a sentence-based MT approach. We show that the use of source-side labels indicating the targeted level of simplification benefited performance. Furthermore, pretraining the encoder and decoder on standard German also improved the performance of the ATS models. In our experiments, we noticed that MT models have a tendency to copy the source segments. While copies are sometimes desirable, we want to avoid this in cases where the original segment could benefit from further simplification.

1The term plain language is avoided here, as it refers to a specific level of simplification. Simplified language subsumes all efforts of reducing the complexity of a text.
We show that the use of special copy labels at training time can positively influence such behavior. In particular, the contributions of the paper at hand are the following:

- We demonstrate the use of source-side Common European Framework of Reference for Languages (CEFR) (Council of Europe, 2009) labels and a fine-tuning approach to boost text simplification performance for certain CEFR levels.

- We investigate the use of source-side copy labels to reduce the copying behaviour of text simplification models in situations where copying is not desirable.

The remainder of this paper is structured as follows: Section 2 describes existing datasets for text simplification for a variety of languages as well as established approaches to ATS. Section 3 describes our approach to multi-level text simplification for German. We discuss our experiments in Section 4 and conclude in Section 5 with further thoughts on improving ATS for German and current challenges to overcome.

2 Previous Work: Automatic Text Simplification

2.1 Data

ATS with sentence-based MT models relies on pairs of standard-language/simplified-language texts aligned at the sentence level. A number of parallel corpora have been created to this end. Gasperin et al. (2010) compiled the PorSimples Corpus consisting of Brazilian Portuguese texts (2,116 sentences), each with two different levels of simplifications (“natural” and “strong”), resulting in around 4,500 aligned sentences. Bott and Saggion (2012) produced the SimpLext Corpus consisting of 200 Spanish/simplified Spanish document pairs, amounting to a total of 1,149 (Spanish) and 1,808 (simplified Spanish) sentences (approximately 1,000 aligned sentences).

A large parallel corpus for ATS is the Parallel Wikipedia Simplification Corpus (PWKP) compiled from parallel articles of the English Wikipedia and the Simple English Wikipedia (Zhu et al., 2010), consisting of about 108,000 sentence pairs. Application of the corpus has been subject to criticism for various reasons (Štajner et al., 2018); the most important among these is the fact that Simple English Wikipedia articles are often not translations of articles from the English Wikipedia. Hwang et al. (2015) provided an updated version of the corpus that includes a total of 280,000 full and partial matches between the two Wikipedia versions.

Another frequently used data collection, available for English and Spanish, is the Newsela Corpus (Xu et al., 2015) consisting of 1,130 news articles, each simplified into four school grade levels by professional editors.

Klaper et al. (2013) created the first parallel corpus for German/simplified German, consisting of 256 texts each (approximately 70,000 tokens) downloaded from the Web. More recently, Battisti et al. (2020) extended the corpus with more parallel data, additional monolingual-only data (in simplified German), and new information on text structure (e.g., paragraphs, lines), typography (e.g., font type, font style), and images (content, position, and dimensions). The corpus is compiled from Web sources in Germany, Austria, and Switzerland. The sources mostly represent websites of governments, specialized institutions, and non-profit organizations. The documents cover a wide range of topics, such as politics (e.g., instructions for voting), health (e.g., what to do in case of pregnancy), and culture (e.g., introduction to art museums). The corpus contains 6,217 documents (5,461 monolingual documents plus 378 documents for each side of the parallel data). The vocabulary of the simplified German texts is smaller than that of the German texts by 51% (33,384 vs. 16,352 types), which is comparable to the rate of reduction reported for the Newsela Corpus (50.8%).

Säuberli et al. (2020) introduced a corpus of news items from the Austria Press Agency (Austria Presse Agentur, APA). At APA, four to six news items per day are manually simplified into two language levels, B1 and A2, following guidelines by capito, the largest provider of simplification services (translations and translators’ training) in Austria, Germany, and Switzerland. These news
items cover the topics of politics, economy, culture, and sports.

A number of tools exist for sentence alignment of parallel documents in the context of sentence simplification; among them are CATS (Štajner et al., 2018), MASSAlign (Paetzold et al., 2017), and LHA (Nikolov and Hahnloser, 2019). Spring et al. (2021) evaluated these alignment methods for German text simplification, together with SBERT (Reimers and Gurevych, 2020) and Vecalign (Thompson and Koehn, 2019). Both of the latter tools were originally designed in the context of multilingual alignment. Evaluation against a human-created gold standard showed that LHA yielded the most accurate sentence alignments.

2.2 Approaches

Specia (2010) introduced statistical machine translation (SMT) to the ATS task, using data from a small parallel corpus (roughly 4,500 parallel sentences) for Portuguese. Coster and Kauchak (2011) used the PWKP Corpus in its original form (cf. Section 2.1) to train an MT system. Xu et al. (2016) performed syntax-based MT on the English/simplified English part of the Newsela Corpus (cf. Section 2.1).

Nisioi et al. (2017) pioneered neural machine translation (NMT) models for ATS, performing experiments with LSTMs on both the Wikipedia dataset of Hwang et al. (2015) and the Newsela Corpus for English, with automatic alignments derived from CATS (cf. Section 2.1).

More recent contributions to ATS include explicit edit operation modeling (Dong et al., 2019), graded simplification (Nishihara et al., 2019), weakly supervised (Palmero Aprosio et al., 2019), and unsupervised approaches (Surya et al., 2019; Kumar et al., 2020).

Suter et al. (2016) introduced a rule-based ATS system for German. Their rules are based on linguistically motivated guidelines and their simplification system yielded outputs with a syntactic complexity comparable to a human translation.

Battisti et al. (2020) presented an approach to German ATS using recurrent neural networks with attention and incorporated back-translation (Sennrich et al., 2016) to generate additional synthetic training data from the monolingual part of their corpus.

Säuberli et al. (2020) presented the first approach to ATS for German using (sentence-based) NMT models. As data, they used the APA Corpus introduced in Section 2.1, amounting to approximately 3,500 sentence pairs.

Other contributions that are relevant to our work originate from the field of MT. Source-side labels have previously been employed in a variety of tasks such as domain adaptation (Kobus et al., 2017), multilingual translation (Johnson et al., 2017), and to improve training with back-translated data (Caswell et al., 2019).

2.3 Evaluation

The most commonly applied automatic evaluation metrics for text simplification are BLEU (Papineni et al., 2002) and SARI (Xu et al., 2016). BLEU, the de-facto standard for automatic evaluation of MT, computes token n-gram overlap between a hypothesis and one or multiple references. A shortcoming of BLEU with respect to ATS is that it does not punish hypotheses that are identical to the input. In contrast, SARI was introduced specifically for ATS and is designed to punish excessive copying behaviour. SARI considers the input and rewards tokens in the hypothesis that do not occur in the input but in one of the references (addition), as well as tokens in the input that are correctly retained (copying) or removed (deletion) in the hypothesis.\footnote{A copy or deletion is considered correct if the token is copied or deleted in at least one of the references.}

Table 1 displays scores for previous sentence-level ATS systems for different languages.

3 Text Simplification Along CEFR Levels for German

3.1 Data

The data used for the experiments reported in this paper consists of two collections:

The first part comprises an expanded version of the Austria Press Agency (Austria Presse Agentur, APA) corpus described by Säuberli et al. (2020) (cf. Section 2.1). Our updated version of this corpus consists of standard-language news items with their corresponding simplifications between August 2018 and April 2021. We extracted simplified German documents along with their standard German counterparts. This extraction yielded 2,410 document pairs for B1 and 2,347 for A2. As human text simplification work at the APA is ongoing, this corpus is expected to grow with time.

The second part of our data consists of the capito corpus. As a provider of simplification services,
capito produces a high number of professional simplifications for a variety of documents and text genres. This includes, but is not limited to, booklets, information texts, websites and legal texts, which are manually simplified into one or more levels following the capito guidelines. The simplification levels in this corpus include B1, A2 and A1. We extracted simplified German documents along with their standard German counterparts and metadata. Currently, the corpus contains 1,245 document pairs for B1, 1,885 for A2 and 879 for A1, however, since capito provides ongoing translation services, the number of documents is constantly increasing.

### 3.2 Sentence Alignment

Sentence alignment for ATS includes some phenomena that do not occur in this form in sentence alignment for translation. Whereas in translation, the standard case is often a simple 1:1 correspondence, alignment for text simplification can be considered n:m, meaning that a single alignment can consist of a varying number of segments on each side. This is due to phenomena such as sentence splitting and compression, additional explanations, as well as the fact that the order of information can change.

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Dataset</th>
<th># Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>OR-B1</td>
<td>APA</td>
<td>10,268</td>
</tr>
<tr>
<td>OR-B1</td>
<td>capito</td>
<td>54,224</td>
</tr>
<tr>
<td>OR-A2</td>
<td>APA</td>
<td>9,456</td>
</tr>
<tr>
<td>OR-A2</td>
<td>capito</td>
<td>136,582</td>
</tr>
<tr>
<td>OR-A1</td>
<td>capito</td>
<td>10,952</td>
</tr>
</tbody>
</table>

The results of NMT experiments are highly dependent on the available data. We extracted sentence alignments from our corpora using the LHA alignment method (Nikolov and Hahnloser, 2019), which was shown to yield the best results for simplified German (Spring et al., 2021) (cf. Section 2.1). For calculating the alignments, we used the Sentence Alignment Tools Evaluation Framework (SATEF),\(^5\) which yields n:m alignments, meaning that a single alignment can consist of a varying number of segments on each side. We aligned our documents in the direction from complex to simple. The number of documents differs considerably

\(^5\) Code is available from: https://github.com/kostrzmar/SATEF
depending on the CEFR level and the dataset, see Section 3.1. Furthermore, the APA Corpus does not contain any data for level A1. This manifests itself in the number of sentence alignments we were able to extract on this level. The largest number of alignments in our parallel corpus are for level A2, followed by about half as many for level B1. With 10,952 sentence alignments, A1 is the simplification level with the smallest amount of data available for model training, see Table 2 for an overview.

The sentence alignments with LHA on the APA data are publicly available.\(^6\)

### 3.3 Text Simplification

All the models we trained for our experiments shared the same architecture and hyperparameters. We trained transformer models (Vaswani et al., 2017) with five layers, four attention heads, 512 hidden units in the transformer layers, and a feed forward layer size of 2048. Embedding dropout and label smoothing were set to 0.3. We used early stopping according to BLEU on a held-out development set with a patience of 10. All models shared a 20k vocabulary between source and target. All our experiments were carried out in sockeye (Hieber et al., 2018).

We trained baseline models where we combined all available training data across all levels. These models had no explicit method to determine the desired level of simplification on the target side.

The diverse dataset allowed us to treat text simplification as a number of subtasks, where the model learns to simplify into different complexity levels, ranging from B1 to A1 according to the CEFR. To allow a model to make a distinction between the different levels of simplification, we used source-side labels indicating the desired CEFR level of the target segment (\(<a1>\), \(<a2>\), and \(<a3>\)). To better understand the copying behavior of our models, we trained our labeled models in two versions: 1) using a simple source-side label indicating the target CEFR level and 2) additionally using an explicit \(<copy>\) label instead of the CEFR level for all segments where source and target were identical. Apart from these modifications to the training data, all model hyperparameters were identical to the baseline models. We will refer to these models as “APA+capito multi” and “APA+capito multi copy”, respectively. Note that copying the source segment to the target is not wrong per se and that there are many cases where no further modification of a segment is needed, especially at higher CEFR levels. We hypothesize that the addition of these copy labels at training time allows the model to better recognize these cases even when they are not present at test time. At test time, all segments were translated with their CEFR label and no \(<copy>\) labels were present. We observed that training a simplification model with explicit \(<copy>\) labels reduced the number of untranslated segments where source and reference are not identical. We treated these types of copies as undesired.

All experiments previously described were performed in two variations. In the first variation, we trained the models from scratch on the simplification data. The second variation involved pretraining the encoder and decoder on a DE→EN or EN→DE translation task, respectively. This was motivated by the relatively low number of aligned segments we could use for our parallel training data. We trained two translation models with the same hyperparameters as the simplification models, but we used separate source and target vocabularies for encoder and decoder (German only). The parallel DE↔EN data for pretraining the NMT models (cf. Section 3.3) consisted of Europarl v10, Common Crawl, News Commentary v15, and the Tilde Rapid Corpus from WMT20.\(^7\) For the simplification models, we then initialized the parameters of the encoder with the encoder parameters of the DE→EN model. Likewise, we initialized the decoder parameters of the simplification models with the decoder parameters of the EN→DE model.\(^8\) The DE→EN, EN→DE and all simplification models used the same German subword vocabulary. We then fine-tuned these pretrained models on our text simplification data. We append the tag “fine-tuned” to the name of these models.

Finally, for the purpose of reproducibility, we trained a labeled simplification model on the publicly available APA alignments\(^9\) described in Section 3.2, referred to as “APA multi”.\(^10\)

For evaluation, we used a test set that consists of

\(^6\)https://zenodo.org/record/5148163

\(^7\)http://www.statmt.org/wmt20/translation-task.html

\(^8\)The cross-attention heads were initialized randomly.

\(^9\)See footnote 6.

\(^10\)Preprocessing and training scripts are available from: https://github.com/ZurichNLP/RANLP2021-German-ATS
500 parallel segments per level (A1, A2 and B1), randomly sampled from the combined corpus. The “APA multi” model was evaluated on a different test set, consisting exclusively of APA data.

3.4 Results

Our results can be found in Tables 3 and 4. In general, the models with target labels for the simplification levels performed better than the baseline both in terms of BLEU and SARI, with the notable exception of BLEU at level A2. Note that A2 was by far the most common simplification level in our dataset. The simple labeling approach of “APA+capito multi” was already effective and outperformed both baseline models on A1 and B1. But it was in turn outperformed by its fine-tuned counterpart “APA+capito multi fine-tuned” on B1, and by the fine-tuned model with copy labels, “APA+capito multi copy fine-tuned”, on A1. Pretraining the level-agnostic baseline model yielded improvements in terms of BLEU for A1 and A2 and only for A2 in terms of SARI.

When evaluating our labeled models with SARI, we could see improvements over the performance of the baselines for all models. Generally, SARI scores suggest that pretraining is especially effective when combined with labeling on levels A2 and B1. On A1, the simple labeling approach of “APA+capito multi” remained the most effective. It yielded an improvement of 6.86 points over the baseline. The best model for A2 and B1 was “APA+capito multi fine-tuned”, which yielded SARI scores improved between 5.49 (A2) and 7.55 (B1).

An analysis of the copying behaviour of the different models can be found in Table 5. The standard fine-tuned model generally had the strongest tendency to copy the source, however, the addition of copy labels significantly reduced the number of copied segments. This was also true for the non-pretrained model variants. Furthermore, the number of undesired copies (where source and reference are not identical) decreased with the use of copy labels (percentage decreases as indicated in Table 5). This was true for both the generic and the pretrained models. In general, the models tended to produce more copies for higher CEFR levels, which was consistent with the training data: In B1, or even A2, shorter segments are more often identical to their standard German counterparts than in A1.

Table 6 shows two examples of how the models with copy labels can avoid source copies. For both samples, “APA+capito multi fine-tuned” simply copies the input. “APA+capito multi fine-tuned copy” avoids this by using two different strategies. In the first example, it produces a segment with different structure and content, which is related to the source segment thematically. Such outputs are common across all models and can be seen as a result from the many-to-many nature of alignment for ATS and the elaborations that are common in text simplification. In the second example, the model produces a shorter simplification by removing some of the information present in the source segment. This ellipsis is another common phenomenon in text simplification.

The performance and copying behaviour of the “APA multi” model cannot be directly compared to the other models because it was trained on different data (exclusively the APA corpus) and uses a different test set.

4 Discussion

Comparing our results to Säuberli et al. (2020), whose experiments are similar to ours in terms of scope and data, it is clear that our baseline models are already quite strong. We find that source-side labels for target language levels generally improve BLEU and SARI scores and that the same is true for pretraining and fine-tuning. Interestingly, combining labels and pretraining results in lower gains in both metrics on A1 for the model without copy labels, indicating that these two approaches cannot always simply be combined. We also note that the scores on CEFR level A2 did not profit as much from the different strategies and we were not able to improve over the pretrained baseline model in terms of BLEU by using labels. We attribute this to the relatively large amount of training data for A2, meaning that specifically dealing with this low resource setting was not needed for this CEFR level. On the other hand, A1 and B2, for both of which there was substantially less data, benefit from labeling and pretraining.

Regarding the copying behaviour, we note higher numbers of direct source copies on the higher CEFR levels. We attribute this to the fact that simplifications on these higher levels typically tend to be closer to the original text in terms of lexical and syntactic complexity. This means that there are more standard language segments without any
Table 3: BLEU scores of the different models. The APA multi model was trained and evaluated on different data and is not comparable to the rest of the models.

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU A1</th>
<th>BLEU A2</th>
<th>BLEU B1</th>
</tr>
</thead>
<tbody>
<tr>
<td>APA multi</td>
<td>15.2</td>
<td>12.3</td>
<td></td>
</tr>
<tr>
<td>Baseline</td>
<td>13.4</td>
<td>14.4</td>
<td>16.3</td>
</tr>
<tr>
<td>Baseline fine-tuned</td>
<td>13.5</td>
<td><strong>14.9</strong></td>
<td>15.7</td>
</tr>
<tr>
<td>APA+capito multi</td>
<td>14.2</td>
<td>14.1</td>
<td>17.2</td>
</tr>
<tr>
<td>APA+capito multi copy</td>
<td>14.0</td>
<td>14.0</td>
<td>15.2</td>
</tr>
<tr>
<td>APA+capito multi fine-tuned</td>
<td>13.9</td>
<td>14.2</td>
<td><strong>17.5</strong></td>
</tr>
<tr>
<td>APA+capito multi copy fine-tuned</td>
<td><strong>14.3</strong></td>
<td>12.4</td>
<td>13.9</td>
</tr>
</tbody>
</table>

Table 4: SARI scores of the different models. The APA multi model was trained and evaluated on different data and is not comparable to the rest of the models.

<table>
<thead>
<tr>
<th>Model</th>
<th>SARI A1</th>
<th>SARI A2</th>
<th>SARI B1</th>
</tr>
</thead>
<tbody>
<tr>
<td>APA multi</td>
<td>42.04</td>
<td>40.73</td>
<td></td>
</tr>
<tr>
<td>Baseline</td>
<td>36.26</td>
<td>36.11</td>
<td>34.53</td>
</tr>
<tr>
<td>Baseline fine-tuned</td>
<td>36.21</td>
<td>36.99</td>
<td>33.98</td>
</tr>
<tr>
<td>APA+capito multi</td>
<td><strong>43.12</strong></td>
<td>41.53</td>
<td>41.81</td>
</tr>
<tr>
<td>APA+capito multi copy</td>
<td>43.11</td>
<td>41.52</td>
<td>40.68</td>
</tr>
<tr>
<td>APA+capito multi fine-tuned</td>
<td>42.88</td>
<td><strong>41.60</strong></td>
<td><strong>42.08</strong></td>
</tr>
<tr>
<td>APA+capito multi copy fine-tuned</td>
<td>42.86</td>
<td>40.86</td>
<td>40.48</td>
</tr>
</tbody>
</table>

5 Conclusion and Outlook

We were able to demonstrate the advantages of different approaches to German multi-level ATS. We established strong baselines on a generic simplification task across all CEFR levels and were able to further boost model performance for specific levels of simplification using source-side labels and a pretraining/fine-tuning strategy. We tested fine-tuning with labeled multi-level models. These approaches were generally more effective on the CEFR levels where we had more limited data, suggesting that they are especially useful in low-resource scenarios. We also investigated the use of copy labels at training time to mark segments where source and target segments are identical. At test time, this resulted in a lower number of copies overall and especially in the number of instances where the references differs from the source. This suggests that copy labels are a valid tool to reduce undesired copying behaviour in text simplification, though their influence on the quality of the output can likely only be determined by human evaluation.

Further work will be conducted on a more advantageous combination of the two approaches of
Table 5: The number of source copies produced by the models. Columns marked with an asterisk only count copies where the source is different from the reference (i.e., undesired copies). The APA multi model was trained and evaluated on different data and is not comparable to the rest of the models.

<table>
<thead>
<tr>
<th>Model</th>
<th>#A1</th>
<th>#A1*</th>
<th>% Undesired A1</th>
<th>#A2</th>
<th>#A2*</th>
<th>% Undesired A2</th>
<th>#B1</th>
<th>#B1*</th>
<th>% Undesired B1</th>
</tr>
</thead>
<tbody>
<tr>
<td>APA multi</td>
<td>4</td>
<td>3</td>
<td>75.00%</td>
<td>2</td>
<td>1</td>
<td>50.00%</td>
<td>75</td>
<td>75</td>
<td>75.76%</td>
</tr>
<tr>
<td>APA-capito multi</td>
<td>58</td>
<td>47</td>
<td>81.03%</td>
<td>60</td>
<td>44</td>
<td>75.00%</td>
<td>99</td>
<td>75</td>
<td>75.76%</td>
</tr>
<tr>
<td>APA-capito multi copy</td>
<td>39</td>
<td>31</td>
<td>79.49%</td>
<td>36</td>
<td>25</td>
<td>69.45%</td>
<td>69</td>
<td>52</td>
<td>75.36%</td>
</tr>
<tr>
<td>APA-capito multi fine-tuned</td>
<td>62</td>
<td>50</td>
<td>80.65%</td>
<td>74</td>
<td>59</td>
<td>79.73%</td>
<td>107</td>
<td>83</td>
<td>77.57%</td>
</tr>
<tr>
<td>APA-capito multi copy fine-tuned</td>
<td>35</td>
<td>27</td>
<td>77.14%</td>
<td>34</td>
<td>24</td>
<td>70.59%</td>
<td>57</td>
<td>42</td>
<td>73.68%</td>
</tr>
</tbody>
</table>

Table 6: Model simplification examples (level A2) comparing the two fine-tuned models with labels in cases where “APA+capito multi fine-tuned” simply copies the source segment. The source is identical to the simplification produced by “APA+capito multi fine-tuned” shown here.

<table>
<thead>
<tr>
<th>Model</th>
<th>German</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source APA+capito multi fine-tuned</td>
<td>Lernen von der Natur!</td>
<td>Learning from nature!</td>
</tr>
<tr>
<td>APA+capito multi fine-tuned copy</td>
<td>Wie funktioniert der Austausch von Wissen?</td>
<td>How does the exchange of knowledge work?</td>
</tr>
<tr>
<td>Source APA+capito multi fine-tuned</td>
<td>Die praktische Fahrradprüfung findet im Grazer Verkehrsgarten im Stadtpark statt.</td>
<td>The practical bicycle test takes place in the Graz traffic garden in the city park.</td>
</tr>
<tr>
<td>APA+capito multi fine-tuned copy</td>
<td>Die praktische Fahrradprüfung findet im Grazer Verkehrsgarten im Stadtpark statt.</td>
<td>The practical bicycle test takes place in the Graz traffic garden in the city park.</td>
</tr>
<tr>
<td>APA+capito multi fine-tuned copy</td>
<td>Die praktische Fahrradprüfung findet im Grazer Verkehrsgarten statt.</td>
<td>The practical bicycle test takes place in the Graz traffic garden.</td>
</tr>
</tbody>
</table>

Acknowledgements

The authors want to thank the Austria Presse Agentur and the CFS GmbH for providing the parallel corpora of standard German documents with their simplified counterparts. The authors are grateful to Marek Kostrzewa for his help in creating the sentence alignments for the corpus and to the anonymous reviewers for their valuable input and feedback. This research was funded by the Austrian Research Promotion Agency (Österreichische Forschungsförderungsgesellschaft, FFG) General Programme under grant agreement number 881202.

References


Gustavo Paetzold, Fernando Alva-Manchego, and Lu-
cia Specia. 2017. Massalign: Alignment and anno-
tation of comparable documents. In Proceedings of
the IJCNLP 2017, Taipei, Taiwan, November 27 - De-
Association for Computational Linguistics.

Alessio Palmero Aprosio, Sara Tonelli, Marco Turchi,
Matteo Negri, and Mattia A. Di Gangi. 2019. Neu-
ral Text Simplification in Low-Resource Conditions
Using Weak Supervision. In Proceedings of the
Workshop on Methods for Optimizing and Evalu-
ating Neural Language Generation, pages 37–44,
Minneapolis, Minnesota.

Sanja ˇStajner, Marc Franco-Salvador, Paolo Rosso, and
Nicolas Spring, Dominik Pfütze, Marek Kostrzewa,
Alessia Battisti, Annette Rios, and Sarah Ebling.
2021. Comparing Sentence Alignment Methods for
Automatic Simplification of German Texts. Presen-
tation given at the 1st International Easy Language
Day Conference (IELD), Germersheim, Germany.

Sanja Štajner, Marc Franco-Salvador, Paolo Rosso, and
Simone Paolo Ponzetto. 2018. CATS: A Tool for
Customized Alignment of Text Simplification Corpo-
ra. In Proceedings of the Eleventh International
Conference on Language Resources and Evaluation
(LREC 2018), pages 3895–3903, Miyazaki, Japan.

Sanja Štajner and Sergiu Nisioi. 2018. A Detailed Eval-
uation of Neural Sequence-to-Sequence Models for
In-domain and Cross-domain Text Simplification.
In Proceedings of the 11th Language Resources and
Evaluation Conference, Miyazaki, Japan.

Julia Suter, Sarah Ebling, and Martin Volk. 2016. Rule-
based Automatic Text Simplification for German.
In Proceedings of the 13th Conference on Natural
Language Processing (KONVENS 2016), pages 279–
287, Bochum, Germany.

Brian Thompson and Philipp Koehn. 2019. Vecalign:
Improved sentence alignment in linear time and
space. In Proceedings of the 2019 Conference on
Empirical Methods in Natural Language Processing
and the 9th International Joint Conference on Natu-val Language Processing (EMNLP-IJCNLP), pages
1342–1348, Hong Kong, China. Association for
Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob
Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz
Kaiser, and Illia Polosukhin. 2017. Attention is all

Andreas Säuberli, Sarah Ebling, and Martin Volk. 2020. Bench-
making data-driven automatic text simplification for German.
In Proceedings of the 1st Workshop
on Tools and Resources to Empower People with
REAding DIfficulties (READI), pages 41–48,
Mar-
seille, France. European Language Resources Asso-
ciation.

Sander Wubben, Antal van den Bosch, and Emiel Krah-
mer. 2012. Sentence Simplification by Monolingual
Machine Translation. In Proceedings of the 50th An-
nual Meeting of the Association for Computational
Linguistics (Volume 1: Long Papers), pages 1015–
1024, Jeju Island, Korea. Association for Compu-
tational Linguistics.

Wei Xu, Chris Callison-Burch, and Courtney Napoles.
2015. Problems in Current Text Simplification Re-
search: New Data Can Help. Transactions of the As-

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze
Chen, and Chris Callison-Burch. 2016. Optimizing
Statistical Machine Translation for Text Simplifica-
tion. Transactions of the Association for Compu-

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych.
2010. A monolingual tree-based translation model
for sentence simplification. In Proceedings of the
International Conference on Computational Linguis-
tics, pages 1353–1361, Beijing, China.
## A Model Hyperparameters

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>architecture</td>
<td>transformer</td>
</tr>
<tr>
<td>seed</td>
<td>1</td>
</tr>
<tr>
<td>patience</td>
<td>10</td>
</tr>
<tr>
<td>optimized metric</td>
<td>BLEU</td>
</tr>
<tr>
<td>batch type</td>
<td>word</td>
</tr>
<tr>
<td>batch size</td>
<td>2048</td>
</tr>
<tr>
<td>update frequency</td>
<td>2</td>
</tr>
<tr>
<td>optimizer</td>
<td>adam</td>
</tr>
<tr>
<td>max length</td>
<td>95:95</td>
</tr>
<tr>
<td>label smoothing</td>
<td>0.3</td>
</tr>
<tr>
<td>vocab</td>
<td>20k</td>
</tr>
<tr>
<td>layers</td>
<td>5:5</td>
</tr>
<tr>
<td>model size</td>
<td>512:512</td>
</tr>
<tr>
<td>heads</td>
<td>4:4</td>
</tr>
<tr>
<td>ff</td>
<td>2048</td>
</tr>
<tr>
<td>dropout attention</td>
<td>0.1</td>
</tr>
<tr>
<td>dropout-act</td>
<td>0.0</td>
</tr>
<tr>
<td>dropout-prepost</td>
<td>0.1</td>
</tr>
<tr>
<td>embedding dropout</td>
<td>0.3</td>
</tr>
<tr>
<td>positional embeddings</td>
<td>fixed</td>
</tr>
<tr>
<td>initial lr</td>
<td>0.0002</td>
</tr>
<tr>
<td>learning-rate-reduce-factor</td>
<td>0.9</td>
</tr>
<tr>
<td>learning-rate-scheduler</td>
<td>plateau-reduce</td>
</tr>
<tr>
<td>init</td>
<td>xavier</td>
</tr>
<tr>
<td>init-scale</td>
<td>3.0</td>
</tr>
<tr>
<td>init-xavier-factor-type</td>
<td>avg</td>
</tr>
</tbody>
</table>

Table 7: Model hyperparameters.
Exploring Reliability of Gold Labels for Emotion Detection in Twitter

Sanja Štajner
Symanto Research
Nuremberg, Germany
sanja.stajner@symanto.com

Abstract

Emotion detection from social media posts has attracted noticeable attention from natural language processing (NLP) community in recent years. The ways for obtaining gold labels for training and testing of the systems for automatic emotion detection differ significantly from one study to another, and pose the question of reliability of gold labels and obtained classification results. This study systematically explores several ways for obtaining gold labels for Ekman’s emotion model on Twitter data and the influence of the chosen strategy on the manual classification results.

1 Introduction

Interest for automatic emotion detection has been gaining popularity in the last ten years (Acheampong et al., 2020, Figures 3 and 4). The span of its applications ranges from empathetic chatbots and virtual agents (Paiva et al., 2017; Rashkin et al., 2019; Lin et al., 2019b; Shin et al., 2019; Lin et al., 2019a; Ma et al., 2020) to social media and public opinion analysis (e.g. (Anstead and O’Loughlin, 2014; Wu et al., 2020; Loureiro and Alló, 2020)). Nevertheless, the task proved challenging, especially when attempted at purely textual utterances as opposed to the multi-modal ones (Poria et al., 2019), probably due to missing visual and audio cues (Acheampong et al., 2020).

Previous studies reported some of the challenges in automatic emotion detection from texts: different perspectives one may take (Buechel and Hahn, 2017b; Alm et al., 2005), missing context (Ôhman et al., 2020; Mohammad, 2012), non-literal meaning (Mohammad, 2012), high subjectivity of the task and low inter-annotator agreement even among trained annotators (Alm et al., 2005; Schuff et al., 2017). For example, the utterance “Italy defeats France in World Cup Final” (Katz et al., 2007) is most probably neutral from the writer’s (journalist’s) perspective, while evoking strong and probably opposite emotions among Italian and French readers (Buechel and Hahn, 2017b). The utterance “Time for shopping” might be neutral, or express evoke various emotions (e.g. joy, anger, fear) depending on the writer’s/reader’s associations and personal experiences with shopping.

The field of emotion detection from text, similar to many other areas of natural language processing, suffers from the absence of standards for human annotation, and systematic investigations of how different strategies for obtaining gold labels influence classification results. Notable exceptions are the studies by Mohammad and Turney (2013) and Buechel and Hahn (2017b). Mohammad and Turney (2013) found that asking annotators which emotion is the word associated with yields higher inter-annotator agreement than asking them which emotion the word evokes. This result indicated that annotating emotions from text’s perspective is less subjective than annotating them from reader’s perspective. Motivated by those results, Buechel and Hahn (2017b) investigated the influence of perspective on the inter-annotator agreement in emotion annotation at a sentence level.

A recent study by Northcutt et al. (2021) demonstrated that incomplete or suboptimal gold labels in benchmark datasets can steer research efforts in wrong direction as they reward systems that comply with such suboptimal labels. Obtaining the correct gold labels for emotion detection from texts should thus be of the utmost importance for the field.

This study’s contributions towards that goal are:

- An overview of previous efforts in human annotation of emotions in texts (Section 2).
- Single-label human annotation of Ekman’s emotions in English tweets by six trained annotators (Section 3).
### Table 1: Annotation procedures used in previous studies ("?" signifies that the particular aspect was not specified in the paper, "+1" in the #emotions column signifies the additional class for "other" or "no emotion").

<table>
<thead>
<tr>
<th>Study</th>
<th>#annotators Per instance</th>
<th>Total</th>
<th>Gold #emotions</th>
<th>Labelling</th>
<th>Perspective</th>
<th>Genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Demszky et al., 2020)</td>
<td>3 or 5</td>
<td>82</td>
<td>&gt; 1 annotator</td>
<td>27+1</td>
<td>multi</td>
<td>writer</td>
</tr>
<tr>
<td>(Bostan et al., 2020)</td>
<td>5</td>
<td>310</td>
<td>&gt; 1 annotator</td>
<td>15+1</td>
<td>single</td>
<td>text</td>
</tr>
<tr>
<td>(Öhman et al., 2020)</td>
<td>≤3</td>
<td>108</td>
<td>&gt; 1 annotator</td>
<td>8+1</td>
<td>multi</td>
<td>speaker</td>
</tr>
<tr>
<td>(Poria et al., 2019)</td>
<td>5</td>
<td>?</td>
<td>majority</td>
<td>6+1</td>
<td>single</td>
<td>speaker</td>
</tr>
<tr>
<td>(Hsu et al., 2018)</td>
<td>5</td>
<td>?</td>
<td>majority*</td>
<td>6+1</td>
<td>single</td>
<td>speaker</td>
</tr>
<tr>
<td>(Schuff et al., 2017)</td>
<td>3–6</td>
<td>6</td>
<td>various</td>
<td>8</td>
<td>multi</td>
<td>?</td>
</tr>
<tr>
<td>(Mohammad et al., 2015)</td>
<td>3+</td>
<td>≈3000</td>
<td>&gt; half</td>
<td>19+1</td>
<td>single</td>
<td>text</td>
</tr>
<tr>
<td>(Brynielsson et al., 2014)</td>
<td>3</td>
<td>3</td>
<td>majority</td>
<td>3+1</td>
<td>single</td>
<td>writer</td>
</tr>
<tr>
<td>(Neviarouskaya et al., 2010)</td>
<td>3</td>
<td>3</td>
<td>≥2 agree</td>
<td>14</td>
<td>single</td>
<td>?</td>
</tr>
<tr>
<td>(Neviarouskaya et al., 2009)</td>
<td>3</td>
<td>3</td>
<td>≥2 agree</td>
<td>9+1</td>
<td>single</td>
<td>?</td>
</tr>
<tr>
<td>(Strapparava and Mihalcea, 2007)</td>
<td>6</td>
<td>6</td>
<td>?</td>
<td>6</td>
<td>multi</td>
<td>reader</td>
</tr>
<tr>
<td>(Aman and Szpakowicz, 2007)</td>
<td>2</td>
<td>4</td>
<td>both agree</td>
<td>6+2</td>
<td>single</td>
<td>text</td>
</tr>
<tr>
<td>(Alm et al., 2005)</td>
<td>2-3</td>
<td>3</td>
<td>majority</td>
<td>6+1</td>
<td>single</td>
<td>text</td>
</tr>
</tbody>
</table>

- Detailed analysis of the collected human annotations and their comparison to the automatically assigned labels that are the current standard for obtaining gold labels on Twitter data (Section 4.1).
- Systematical investigation of several strategies for obtaining gold labels from manual annotations, and their influence on the reported manual classification results (Section 4.2).

### 2 Related Work

Several recent surveys (Acheampong et al., 2020; Alswaidan and Menai, 2020) and studies (Öhman et al., 2020; Bostan et al., 2020; Bostan and Klingler, 2018; Schuff et al., 2017) list previous work on emotion detection from texts and emphasise their differences in type of emotion taxonomy, task (single-label or multi-label), size of the dataset, text genre, granularity, topics, system architectures, and best results obtained with systems for automatic detection of emotions in texts.

However, none of the studies focussed on assessing the quality of benchmark datasets, or the influence of methods used for obtaining gold labels on the results of systems for automatic emotion detection from texts.

Drawing conclusions about influence of strategies for obtaining gold labels on the classification results by systematic exploration of the previous studies is not possible due to different text genres, number and type of annotators (trained vs. crowdsourced), annotation type (single-label vs. multi-label), granularity of annotations (word or sentence level, with or without surrounding context), emotion taxonomies, and the perspective taken. Table 1 presents annotation strategies used in some of the previous studies. For instance, in a multi-labelling task with 27 emotions (+ neutral) where each Reddit comment was annotated by three or five annotators out of a total of 82 crowdsourced annotators, the Cohen’s kappa (Cohen, 1960) was calculated by aggregating all pairs of annotations per instance and emotion (Demszky et al., 2020). In a single-labelling task with 15 emotions (+ no emotion) where each news headline was annotated by five out of 310 annotators, in contrast, the authors report the Fleiss’ kappa (Fleiss, 1971) as a measure of inter-annotator agreement (Bostan et al., 2020). In the XED dataset of movie subtitles, annotated with 8 emotions (+ neutral) in a multi-labelling task, some instances were annotated with fewer than three annotators (some even one annotator only), and the Cohen’s kappa was calculated for gold labels in the parallel dataset of movie subtitles in English and Finish (Öhman et al., 2020).

Some studies went beyond the “simple” emotion annotation, by requesting from annotators to annotate the intensity of the emotions, e.g. (Strapparava and Mihalcea, 2007; Aman and Szpakowicz, 2007; Buechel and Hahn, 2017a), the triples of who experiences which emotion and why (Kim and Klinger, 2018). In a recent study, Bostan et al. (2020) conducted the annotation of emotions, cues, intensities, experiencers, causes, targets, and reader’s emotions on news headlines.

A commonly used strategy for obtaining large training datasets for emotion detection in texts is
by automatically labelling tweets that contain hashtags that explicitly mention emotions or predefined emotion keywords. Wang et al. (2012) used 131 emotion hashtags as keywords for collecting 5 million tweets in seven emotion categories (joy, sadness, anger, love, thankfulness, surprise). They explored several different strategies for obtaining gold labels based on hashtags and found that most accurate gold labels are obtained if the keyword hashtag appears at the end of the tweet; keyword hashtags appearing anywhere else in the tweet were not found to be that relevant. Shahrai and Zaiane (2017) automatically annotated tweets with nine emotions (anger, fear, joy, love, sadness, surprise, thankfulness, disgust and guilt) based on 15 explicit hashtags appearing in them, resulting in Clean Balanced Emotional Dataset (CBET) with 27,000 annotated tweets (3,000 per each emotion). Mohammad (2012) compiled a corpus of 21,051 tweets which contained one of the six Ekman’s emotions (anger, disgust, fear, joy, sadness, surprise) as the last hashtag, and suggested to use it as an automatically assigned label that corresponds to the emotion experienced by the writer. One of the findings of that study was that such large, automatically-labelled training dataset can be used for emotion detection in other domains and text genres.

3 Methodology

3.1 Dataset

The dataset used in this study is a subset of TEC dataset (Mohammad, 2012), available through the Unify dataset (Bostan and Klinger, 2018). It consists of 35 randomly selected English instances for each of the six Ekman’s emotions (anger, disgust, fear, joy, sadness, and surprise) that contain at least six tokens. To select 35 instances of each of the six emotions, the gold labels of the original dataset were used, i.e. the automatically-assigned gold labels based on the last hashtag in the post, as described in the previous section, e.g. “We are fighting for the 99% that have been left behind. #OWS #anger” (Mohammad, 2012).

3.2 Annotation Procedure

Annotators. Each of the 210 tweets (35 per each emotion class) was annotated by six trained, paid annotators. Three annotators were male and three female. All annotators had at least a bachelor degree. Two of the annotators (one male and one female) were native speakers of English (UK); the other four annotators were proficient in English, and use it in their everyday work.

Guidelines. The annotators were instructed to choose, for each tweet, from a drop-down menu, one of the seven possible labels (ANGER, DISGUST, FEAR, JOY, SADNESS, SURPRISE, NEUTRAL), which best represents the emotion of the writer of the post, as the automatically assigned gold labels were expected to represent writers’ emotions as well (Mohammad, 2012).

Evaluation. The inter-annotator agreement, either between the gold label and one of the annotators, or between two annotators, was calculated in two ways: as accuracy (the percentage of cases in which the given labels match); and Cohen’s $\kappa$ (Cohen, 1960).  

3.3 Experiments

Two sets of experiments were conducted. In the first set of experiments, the results of the human annotation experiment were analysed and compared with the automatically obtained gold labels. The goals of this set of experiments were: (1) to investigate reliability of automatically obtained gold labels; and (2) to estimate complexity of the task for trained human annotators and find the main causes of their disagreements.

In the second set of experiments, several strategies for obtaining gold labels from human annotations, and their influence on the manual classification results were explored, given that previous studies used various strategies for obtaining gold labels from human annotations (Table 1, Section 2). The explored strategies for obtaining gold labels were: (1) based on the last emotion-hashtag (no human annotation required); (2) based on the annotations of just one trained annotator; (3) based on the majority label obtained from three human annotations; and (4) based on the majority label obtained from five human annotations.

4 Results and Discussion

The results of the two sets of experiments are presented and discussed in two separate subsections.

4.1 Annotation Analysis

The pairwise inter-annotator agreements for each pair of annotators, and for each annotator and the

\footnote{For calculation of Cohen’s $\kappa$, we used the implementation in sklearn library for python: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.cohen_kappa_score.html.}
Table 2: Statistics of the pairwise inter-annotator agreement

<table>
<thead>
<tr>
<th>Annotation pair</th>
<th>Cohen’s κ</th>
<th>Agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
<td>max</td>
</tr>
<tr>
<td>Two annotators</td>
<td>0.33</td>
<td>0.55</td>
</tr>
<tr>
<td>Gold vs. annotator</td>
<td>0.13</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Table 2: The percentage of instances on which certain number of annotators assigned the gold label.

<table>
<thead>
<tr>
<th>Gold emotion</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANGER</td>
<td>31.4%</td>
<td>14.3%</td>
<td>5.7%</td>
<td>14.3%</td>
<td>5.7%</td>
<td>17.1%</td>
<td>11.4%</td>
</tr>
<tr>
<td>DISGUST</td>
<td>37.1%</td>
<td>20.0%</td>
<td>5.7%</td>
<td>8.6%</td>
<td>5.7%</td>
<td>8.6%</td>
<td>14.3%</td>
</tr>
<tr>
<td>FEAR</td>
<td>71.4%</td>
<td>17.1%</td>
<td>5.7%</td>
<td>2.9%</td>
<td>2.9%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>JOY</td>
<td>42.9%</td>
<td>17.1%</td>
<td>8.6%</td>
<td>17.1%</td>
<td>5.7%</td>
<td>0.0%</td>
<td>8.6%</td>
</tr>
<tr>
<td>SADNESS</td>
<td>25.7%</td>
<td>25.7%</td>
<td>11.4%</td>
<td>5.7%</td>
<td>11.4%</td>
<td>11.4%</td>
<td>8.6%</td>
</tr>
<tr>
<td>SURPRISE</td>
<td>45.7%</td>
<td>22.9%</td>
<td>5.7%</td>
<td>14.3%</td>
<td>5.7%</td>
<td>2.9%</td>
<td>2.9%</td>
</tr>
<tr>
<td>all</td>
<td>42.4%</td>
<td>19.5%</td>
<td>7.1%</td>
<td>10.5%</td>
<td>6.2%</td>
<td>6.7%</td>
<td>7.6%</td>
</tr>
</tbody>
</table>

Gold labels, are presented in Table 2. As can be seen, the minimum, maximum, and average pairwise inter-annotator agreement is notably lower between any annotator and the gold labels than between any two annotators. The Cohen’s κ score between any annotator and the gold label corresponds to only a slight (strength of) agreement (0.00 ≤ κ ≤ 0.20), while the Cohen’s κ for any pair of annotators range from fair (0.21 ≤ κ ≤ 0.40) to moderate (0.41 ≤ κ ≤ 0.60) agreement, according to Landis and Koch (1977).

To investigate whether the annotation disagreements stem from poor quality of annotations, or from the natural complexity of the task, all labels assigned by all six annotators were manually checked. It was found that none of the annotators had more than 1% of erroneous annotations. The found errors were either due to labelling topic of the post instead of the writer’s emotion (SURPRISE labels), or due to labelling posts based on the words that occur in them instead of the writer’s emotion (SADNESS labels). The rest of the annotation disagreements were the result of the natural complexity of the task (Section 4.1.2).

The fairly low agreement among the annotators indicates that the task of detecting Ekman’s emotions in tweets is challenging and/or subjective. This is in line with previous studies which reported that emotion detection from text is a complex task that results in low inter-annotator agreements regardless of the emotion taxonomy used (Alm et al., 2005; Schuff et al., 2017; Kim and Klinger, 2018; Bostan and Klinger, 2018; Öhman et al., 2020; Acheampong et al., 2020).

The very low agreement between the annotators and the gold labels, in turn, indicates potential problems with the strategy of automatically assigning gold labels for emotion in tweets (according to the last hashtag of the tweet). This is in line with the results reported by Demszky et al. (2020) where the transfer-learning based system obtained noticeably lower results on the TEC dataset (F1-score of ≈0.5) than on the other two Twitter datasets where the gold label were obtained by manual annotations (F1-scores of ≈0.8 and ≈0.7).

4.1.1 Reliability of the Gold Labels

To explore the main causes of disagreements of the annotators with the automatically assigned gold labels, the percentage of instances on which certain number of annotators assigned the same label as the gold one was calculated for each emotion category separately (Table 3). In as many as 42.4% of the cases, none of the six trained annotators assigned the same label as the gold one was calculated for each emotion category separately (Table 3). In as many as 42.4% of the cases, none of the six trained annotators assigned the same label as the gold one. All six annotators assigned the same label as the gold one in only 7.6% of the cases. While the latter can be a consequence of a high subjectivity of the task, the former indicates that the automatically assigned gold labels might not be reliable.

In the per-class analysis (Table 3), FEAR and SURPRISE stand out as gold labels for which in as many as 71.4% and 45.7% of the cases, respectively, none of the six annotators assigned the gold
# Example Gold Assigned
1 Relatives here. Hafta sleep on a couch in the basement. #cantsleep #effuguysiwantmyqueensize
2 There is dirty underwear on the floor of the Men’s room in Dillons.
3 Sometimes in life u just have to DO IT. holds people back from doing so many things!
4 Courage is the path that leads from to action. Christian McCor-mack #quote #quotes
5 My team is starting to heat up you can’t contain us too long let the blowout begin ducks attack the duck
6 Wanna be remembered? On black friday, go to a store, push a kid over, look him in the eye and say “You remember me”
7 I like doing stuff for my close friends when they don’t expect it for @lexi_peters
8 Looking forward to get this done and seeing the reaction from my beautiful gf if you ask I won’t say what it is #happy

Table 4: Examples of complete disagreement of annotators with the gold labels FEAR and SURPRISE. The number in parenthesis after the label signifies the number of annotators who assigned that label.

Table 5: Most frequently assigned sets of labels.
other disagreements in that group were between ANGER and SADNESS, ANGER and DISGUST, and JOY and SURPRISE (Table 5).

The most surprising combinations, among the most frequently encountered ones, were \{JOY, SADNESS\} (in 2.4% of all instances) and \{JOY, SADNESS, NEUTRAL\} (in 4.8% of all instances). By manual inspection of all those surprising combinations, it was discovered that they were not a result of erroneous annotations, but were rather assigned to one of the two following types of posts: (1) posts that can be associated with either joy or sadness depending on the writer’s association with the mentioned action (examples 1 and 2 in Table 6); (2) posts that contain one part that conveys writer’s sadness and other that conveys writer’s joy (examples 3–5 in Table 6). Disagreements that stem from the first type of posts cannot be avoided. Disagreements that stem from the second type of posts, in contrast, could be avoided by more fine-grained emotion annotation where annotators also mark the causes for each found emotion in the sentence, as it was done in studies by Kim and Klinger (2018) or Bostan et al. (2020), mentioned in Section 2.

### 4.2 Comparison of Different Strategies

To systematically analyse the influence of different strategies for obtaining gold label on the manual classification performances, the Cohen’s \(\kappa\) and accuracy of each remaining annotator (whose annotations were not used for obtaining the gold labels) against the gold labels were calculated. As the focus of this study is on single-label classification task, the majority vote was used as the gold label (other strategies mentioned in Table 1 would lead to multiple gold labels per instance). As mentioned in Section 3.3, four strategies were explored: automatically obtaining gold labels based on the last hashtag (0 annotations), having one human annotator to provide the gold label (1 annotation), having three human annotators to provide the gold label as the majority label (3 annotations), and having five human annotators to provide the gold label as the majority label (5 annotations). In the last two strategies, it is not always possible to obtain the majority label, i.e. if all three annotators assign different labels (for the strategy where gold labels are obtained based on three human annotations), or if two distinct labels were assigned each by two annotators (for the strategy where gold labels are obtained based on five human annotations). The minimal, maximal, and average percentage of instances (out of 210) that did not have a majority class are given in Table 7.

Cohen’s \(\kappa\) and accuracy were computed for each annotator against the gold labels only for those annotators whose annotations were not used for obtaining the gold labels, e.g. if the gold labels were obtained by using annotations of the annotators 1, 2, and 5, then the Cohen’s \(\kappa\) and accuracy against the gold labels were calculated only for the annotators 3, 4, and 6. That resulted in 20 data points for the strategy of obtaining gold labels from three human annotations (all combinations of three annotators from a total of six annotators), and six data points for the strategies of obtaining gold labels from one or five human annotations. The cases without majority class (when obtaining gold label from three and five annotations) were excluded.

The influence of the strategy for obtaining gold labels (involving zero, one, three, or five annotators) on the observed manual classification performances (Cohen’s \(\kappa\) and accuracy) of the rest of the annotators is clearly visible in Figure 2. As the quality of annotations was previously manu-

<table>
<thead>
<tr>
<th>#</th>
<th>Example</th>
<th>Assigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Another evening, another cup of coffee.</td>
<td>NEUTRAL(3), SADNESS(2), JOY(1)</td>
</tr>
<tr>
<td>2</td>
<td>At the dentist bright and early</td>
<td>JOY(3), NEUTRAL(2), SADNESS(1)</td>
</tr>
<tr>
<td>3</td>
<td>No school, getting up at 8 for a seven hour car ride at least i have #noschool</td>
<td>SADNESS(3), JOY(3)</td>
</tr>
<tr>
<td>4</td>
<td>Finally done with work and have to be back in less than 12 hours</td>
<td>SADNESS(5), JOY(1)</td>
</tr>
<tr>
<td>5</td>
<td>The movie click is old but one of my favs the ending when he dies makes me tear up</td>
<td>SADNESS(5), JOY(1)</td>
</tr>
</tbody>
</table>

Table 6: Examples that were assigned both JOY and SADNESS. The number in parenthesis after the label signifies the number of annotators who assigned that label.

<table>
<thead>
<tr>
<th>Annotations for gold</th>
<th>Cases without majority</th>
<th>Data points</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>11.4%</td>
<td>22.9%</td>
</tr>
<tr>
<td>5</td>
<td>11.4%</td>
<td>16.2%</td>
</tr>
</tbody>
</table>

Table 7: Percentage of instances without majority class.
Figure 2: Influence of the strategy for computing gold label on the classification performance.

Figure 3: Influence of the strategy for computing gold label on the Cohen’s kappa score between the gold label and the rest of the annotators.

Figure 4: Influence of the strategy for computing gold label on the accuracy of the rest of the annotators.

ally checked (Section 4.1), the higher classification performances in Figure 2 (the upper right corner of the plot) correspond to better quality of gold labels. According to these results, assigning gold labels based on the majority vote of three or five annotators lead to noticeably higher quality than assigning them based on the annotations of one annotator only. Assigning gold labels automatically, based on the emotion explicitly mentioned in the last hashtag in the tweet, leads to significantly lower quality than any other strategy. Box-plots in Figures 3 and 4 show more detailed results of this analysis. Strategies for obtaining gold labels from three or five human annotations both result in moderate to high inter-annotator agreements according to Cohen’s $\kappa$ score, and similar average values of those agreements.

The results of this analysis indicate that for obtaining good quality gold labels for Ekman’s emotions on English Twitter data it is not necessary to hire more than three trained human annotators. Would the same hold for crowdsourced annotations and different emotion taxonomies is something that needs to be explored in future studies.

5 Final Discussion and Conclusion

This study addressed the issue of reliability of gold labels for emotion detection in English tweets.

The results indicated that automatically obtained gold labels (based on the last emotion-hashtag of the tweets) are not reliable, mainly due to the last emotion-hashtag often representing either the topic of the post and not the emotion experienced by the writer, or the emotion that the post is expected to evoke in a particular group of readers. These results call for caution if such large automatically annotated datasets are used for training automatic emotion detection systems, or for testing them, as a significant portion of instances contain suboptimal gold labels.

The analysis of most common disagreements among the annotators revealed that, surprisingly, joy and sadness are often assigned to the same post by different annotators. A manual inspection of those cases revealed that they are results of either lack of context and knowledge about writer’s position about a certain topic, or writer’s expression of both sadness and joy in different parts of the post.

The analysis of impact of strategy used for obtaining gold labels on the manual classification results and quality of the test dataset indicated that three trained annotators are sufficient for providing gold labels by their majority vote.

Ethics/Impact Statement

This study is expected to have a broader impact on the field of automatic emotion detection in texts by raising awareness about the complexity of the task, and encouraging other NLP researchers to further explore annotation procedures and the qual-
ity of the gold labels in datasets used in automatic emotion detection.

The use of suboptimal annotation schemes and procedures, that do not account for natural complexities of the task, may lead to a high number of incorrect or incomplete labels in compiled datasets. The use of such datasets to train and test NLP models further leads to rewarding the models which are not actually performing well on the final goal but are, instead, good at learning and propagating the errors found in the training datasets (Northcutt et al., 2021). This might be particularly dangerous in the case of automatic emotion detection, as such models might be used in the real-world scenarios for a direct communication with real users, e.g. in empathetic chatbots. If those systems fail to grasp the actual emotional state of the user, especially in the case of individuals who are at-risk conditions in terms of mental health, they may cause further harms for such users. Furthermore, apart from the accurate recognition of the emotional experiences, such systems need to adequately respond to those users that go through emotional upheavals. Thus, special attention should be paid in the development of proper empathetic reactions of chatbots to prevent the potential harm to vulnerable populations.

Acknowledgements

I would like to express my gratitude to Justyna Hedler for her help in conducting human annotation experiment, Seren Yenikent for valuable discussions about emotion detection in psychology, Francisco Rangel for supporting this work, and the six annotators for the high quality of annotations.

References


Maria L. Loureiro and Maria Alló. 2020. Sensing climate change and energy issues: Sentiment and emotion analysis with social media in the u.k. and spain. Energy Policy, 143:111490.


Alena Nevirouskaya, H. Prendinger, and M. Ishizuka. 2009. Compositionality principle in recognition of fine-grained emotions from text. In FCWSM.


How to Obtain Reliable Labels for MBTI Classification from Texts?

Sanja Štajner
Symanto Research
Nuremberg, Germany
sanja.stajner@symanto.com

Seren Yenikent
Symanto Research
Nuremberg, Germany
seren.yenikent@symanto.com

Abstract

Automatic detection of the Myers-Briggs Type Indicator (MBTI) from short posts attracted noticeable attention in the last few years. Recent studies showed that this is quite a difficult task, especially on commonly used Twitter data. Obtaining MBTI labels is also difficult, as human annotation requires trained psychologists, and automatic way of obtaining them is through long questionnaires of questionable usability for the task. In this paper, we present a method for collecting reliable MBTI labels via only four carefully selected questions that can be applied to any type of textual data.

1 Introduction

The Myers-Briggs Type Indicator (MBTI) model (Briggs-Myers and Myers, 1995) is one of the most widely used non-clinical psychometric models (Štajner and Yenikent, 2020). It classifies people into two groups across four dimensions: extraversion/introversion (E/I), sensing/intuition (S/N), thinking/feeling (T/F), and judgement/perception (J/P). This leads to a total of 16 personality types. The first three dimensions are based on the theoretical work of Carl Jung (1921), while the fourth dimension was added later by Myers and Briggs-Myers (1995). The MBTI personality framework has already been used for decades in educational and industry settings, e.g. for finding jobs that best resonate with the person’s preferences for information processing (S/N and T/F dimensions), finding work organization types that best resonate with the person’s preferred judgement processes (J/P dimension) thus leading to better job satisfaction, and for better matching work environments with the person’s preferences (E/I dimension) to lower employee turnover (Briggs-Myers and Myers, 1995).

The original MBTI questionnaire contains 93 questions and is not freely available.1 Due to the popularity of MBTI framework (it is estimated that more than 2 million US adults complete the inventory every year),2 there is a number of freely available alternative MBTI questionnaires on the internet, with the 16personalities test3 being one of the most popular ones. According to the Myers-Briggs Foundation4 and the 16personality test website,5 both questionnaires satisfy the accepted standards for test validity and reliability. Nevertheless, the MBTI questionnaires have received a noticeable criticism from the academic community (Pittenger, 1993; Boyle, 1995) for not relying on a scientifically proven (i.e. data-driven) background, but rather on qualitative measures such as observation and introspection. The other common criticism is the binary nature of the questionnaire as it is known that the majority of people usually lies somewhere in the middle of the scales (Pittenger, 1993).

The questionnaire-based personality detection has several weaknesses: it requires trained human assessors; it is prone to social desirability bias (Krumpal, 2011) and reference-group effect (Heine et al., 2002); it is questionable if answering questionnaires is a natural way of showing ones personality (as opposed to free writing or behaviour “when nobody watches”). To detect MBTI typologies in a more natural way and without necessity for trained human assessors, many studies have attempted at building systems for automatic detection of MBTI personality types from text in the last several years. Attempts have been made for automatic detection of MBTI personality types from: tweets written in English (Plank and Hovy, 2015), six other Western European languages (Ver-professional/versions-of-the-mbti-questionnaire/)

1https://www.myersbriggs.org/using-type-as-a-
2https://www.verywellmind.com/the-myers-briggs-type-indicator-2795583#the-mbti-today
3https://www.16personalities.com/free-personality-test
4myersbriggs.org
5https://www.16personalities.com/articles/reliability-and-validity
hoeven et al., 2016), and Japanese (Yamada et al., 2019); English posts collected from Personality Cafe forum\(^6\) available in Kaggle;\(^7\) and English Reddit comments (Gjurković and Snajder, 2018; Gjurković et al., 2020). Despite being trained on large amounts of textual data (over one million), and modelled as four binary classification tasks, the best systems performed only slightly better than the random and majority-class baselines, regardless of the architecture used.

Some studies suggested that tweets might not contain sufficient amounts of MBTI signals (even after concatenating up to 150-200 tweets per user) due to the nature of Twitter posts (Celli and Lepri, 2018; Štajner and Yenikent, 2020, 2021). Another issue with all those studies and obtained results might be that the systems are supervised and were trained with gold labels obtained via MBTI questionnaires that suffer from all earlier mentioned weaknesses. In our recent study (Štajner and Yenikent, 2021), we found a low association between the MBTI types obtained via questionnaires and the MBTI signals found in the short texts written by participants (tweets and free texts on carefully chosen topics). At the same time, the inter-annotator agreement of two expert annotators assigning MBTI types based on those free texts was quite high (Štajner and Yenikent, 2021).

**Contributions.** To avoid all previously mentioned problems in automatic MBTI detection from texts, in this study, we propose a carefully designed set of four questions with answers on a 1-5 scale (Section 3) that aim to capture the main MBTI characteristics without taking much time from participants, and can be administered together with any open-end questions without need for trained human assessors. The validity of our questionnaire has been assessed via expert human annotation following previously proposed annotation methodology (Štajner and Yenikent, 2021). The agreement between the answers to the newly proposed questions and the expert human annotations was found to be similar as between two trained annotators (Section 5.2). Another advantage of the proposed method is that it goes beyond binary typology, by offering a 5-point scale for each MBTI dimension. This creates a possibility for filtering out those instances written by people who exhibit similar amount of signals from both polarities. As it is known that many people have characteristics of both polarities across MBTI dimensions (Pittenger, 1993), such filtering of training datasets might lead to better performances of automatic systems for MBTI detection from texts by removing noise.

2 Related Work

Plank and Hovy (2015) were the first to explore the use of Twitter data for obtaining a large-scale dataset for open-vocabulary automatic detection of MBTI personality traits. They collected a corpus of 1.2M English tweets automatically labelled for gender and MBTI type. To identify the users for whom an MBTI type can be automatically assigned, the authors relied on mentions of any of the 16 MBTI types plus the word “Briggs”. Additionally, each user was labelled as female or male whenever it was discernible; those users for whom the gender was not discernible were excluded from the study. For each selected Twitter user, the authors collected up to 2000 most recent tweets (to be included, each user had to have at least 100 tweets). Plank and Hovy (2015) found that the distribution of MBTI types across the selected Twitter users significantly differs from the distribution of MBTI types across the general US population. The authors further trained binary classification models (for each MBTI dimension separately) using various features and model architectures. The best systems outperformed majority-class baselines only for I/E and T/F dimensions.

Verhoeven et al. (2016) used a similar strategy for obtaining large-scale MBTI datasets for six other languages: German, Italian, Dutch, French, Portuguese, and Spanish. As opposed to the work of Plank and Hovy (2015), the triggers for identifying users whose MBTI types can be automatically assigned were mentions of one of the 16 personality types and the word “personality” or pronouns and verb forms such as “I am” or “I have”, for each of the six languages. All retrieved contexts were manually checked for whether or not they describe the personality of the writer of the post. For all users whose posts passed this check, the gender was annotated based on the user’s name, handle, description, and profile picture (Verhoeven et al., 2016). Distributions of MBTI types across Twitter users of the six languages were found to be similar, with only a few exceptions (Verhoeven et al., 2016). The authors also trained binary classifiers using the dataset with 200 concatenated tweets for each user

---

\(^6\)https://www.personalitycafe.com/

\(^7\)https://www.kaggle.com/datasnaek/mbti-type
and LinearSVC classifier with binary word and character n-gram features. Similar as for English (Plank and Hovy, 2015), in most of the languages, the best classifiers outperformed the majority-class baselines only for E/I and T/F dimensions.

Gjurković and Šnajder (2018) compiled a large-scale MBTI dataset from English Reddit comments by relying on flairs—short introductions of users on various subreddits—which, in the case of the MBTI-related subreddits, usually contain the users’ MBTI results. In the subsequent study (Gjurković et al., 2020), dataset was further enriched with demographic information about the users (age, gender, location, and language), and the labels for two other personality models. The distribution of MBTI types in this dataset also significantly deviated from the general US population (see Figure 3 in Section 6 for comparison of MBTI type distribution among different populations/datasets).

Automatic assignment of MBTI type to each user in all above-mentioned studies is based on automatic extraction of contexts in which a certain MBTI type is mentioned. Without manual inspection of each such mention—which was only reported for the study by Verhoeven et al. (2016)—the assigned labels might not be reliable, as they may refer to someone else mentioned in the tweet and not the writer of the tweet, or they might be a part of a larger phrase, e.g. “I think/believe I am an INTP” or “I expect to get ESFJ as the result if I do personality assessment”.

To the best of our knowledge, the only study in which MBTI labels were obtained by explicitly asking participants to report their MBTI type, if they had done an MBTI personality test in the past, is our recent study (Štajner and Yenikent, 2021). The Amazon Mechanical Turk workers were also asked to describe their favourite type of vacations and preferred hobbies in minimum 300 characters each. We found that this type of texts (responses to carefully selected open-end questions) contain more MBTI signals than tweets (even if concatenated together for each user). We further proposed detailed guidelines for MBTI personality annotation from textual data, and showed that expert human annotators have a high level of agreement among themselves on obtained textual answers when following provided guidelines. At the same time, we found that the annotators have a low level of agreement with the MBTI types reported by participants (based on their previous MBTI personality testing via popular questionnaires), which might be an indication that MBTI results obtained via questionnaires do not resonate well with the MBTI signals found in more natural textual forms.

The current study aims to overcome previously reported issues by proposing four questions with the answers on a 1–5 scale to obtain MBTI labels that better resonate with the expert human MBTI annotations on short texts.

3 Questionnaire

The whole questionnaire consisted of one optional question “You might have obtained your MBTI type in the past via questionnaires. If you know your MBTI type, please type it here”, four compulsory demographic questions, four compulsory questions with answers on a 1–5 scale that aimed to capture the participants MBTI type, and two compulsory open-end questions. Demographic questions encompassed gender, age, whether or not English is their native language, and the highest level of education obtained (Figure 1). The gender question had four possible answers: female, male, other, prefer not to specify. Five age groups were offered to choose from: 18–25, 26–35, 36–45, 46–55, and over 55.

After answering demographic questions, parti-
pants were provided with four questions that aimed to capture their MBTI type, and were asked to provide an answer on a 1–5 points scale. Those four questions are the central contribution of this study. By following the idea that aspects of leisure time represent the most natural version of personality, as it is directed by high degrees of intrinsic motivation (Štajner and Yenikent, 2021), the questions are focused on typical leisure time activities—hobbies and vacations. This also gave us the opportunity to utilize the previously proposed open-end questions (Štajner and Yenikent, 2021) in the validation process (Section 5). In deciding the content of the questions for each individual dimension, we followed the main definitions provided by Briggs-Myers and Myers (1995). Although each MBTI dimension corresponds to multiple practical and behavioral characteristics, the core theoretical focus for every dimension is consistent.

The first question (for the E/I dimension) was designed with the idea of capturing whether the person prefers to be surrounded by people and social interactions, on one end of the scale (1 = extraverted), or to spend quiet and calm time by themselves, on the other end of the scale (5 = introverted). The second question (for the S/N dimension) aims to capture the characteristics of the tasks people would prefer to process, concrete or intuitive, by asking whether they prefer technical and hands-on hobbies (1 = sensing) or abstract and imaginative (5 = intuitive). The third MBTI dimension (T/F) is fundamentally about how people make their decisions, whether based on rational or emotional motives. As people do not engage with strict decision-making processes during their free time, which is ultimately based on their personal interests, the question measured the preference for rational (1 = thinking) or emotional (5 = feeling) reasoning for liking a certain hobby. The fourth question aimed to capture the preference for spontaneous and flexible (1 = perceiving), or a well-planned (5 = judging) schedule at vacations.

We initially prepared two questions per each MBTI dimension and performed a pilot study with 30 participants to choose those questions (Figure 2) that better correspond to the MBTI types provided by the participants, and the MBTI annotations by two annotators.

Finally, participants were asked to answer to two open-end questions, which we previously proposed (Štajner and Yenikent, 2021) as the optimal questions for annotating MBTI types from texts:

- Describe which kind of vacations you typically enjoy and why.
- Describe what type of hobbies you enjoy and why.

The two questions were preceded by the following instructions: “The following questions aim to understand your life style preferences. While answering, please write down the first things that come to your mind without much contemplation.” To be accepted, each answer needed to contain a minimum of 300 characters.

4 Challenges in Data Collection

Data was collected via Amazon Mechanical Turk (AMT) platform. We prepared the questionnaire as Google Forms and provided the link to it in the HIT of the AMT platform. We experimented with various setups in the platform: different values for monetary compensations, allowing only those participants with high scores on previous tasks, different times for validation of the answers and payment. The only variable that noticeably influenced the time needed for obtaining completed HITs was whether or not we restrict the participants according to their performance on the previous HITs. Without any restrictions, we were
obtaining approximately 50 completed HITs per hour. The main bottleneck in the whole procedure was the need for manually checking “honesty” of the answers to the open-end questions.

We manually checked all answers by particularly focusing on checking whether or not: (1) all personality questions contained the same answer (although theoretically possible that a person has the same answer to all four questions, it is very likely that this behaviour instead signifies that the AMT worker just wanted to finish the task as soon as possible and get the monetary compensation); (2) the answers to the open-end questions make sense, i.e. are not just a random sequence of 300 characters; (3) the answers to the open-end questions are copied from the internet; (4) the same worker has already completed the task, and if they did, we checked whether the answers to both multiple-choice and open-end questions were similar in both completed HITs. We found that approximately one third of the completed HITs contained answers that were copied from the internet. Those HITs were disregarded and those workers did not obtain monetary compensation to prevent them from doing it again. In those cases where more than one HIT was completed by the same worker, if the answers were similar, we paid the monetary compensation for all of them and maintained only one randomly chosen HIT for our dataset. In those cases where the answers in the HITs of the same worker had many significant differences, we paid the monetary compensation, but excluded all HITs from that worker from the dataset.

5 Validation

We performed two types of validation of our questionnaire. First, we calculated the agreement of the answers to our personality questions with the MBTI labels provided by the participants, for those cases where the MBTI label was provided (Section 5.1). Given that the MBTI labels obtained by using popular questionnaires might not be reliable for MBTI type detection from textual utterances (Štajner and Yenikent, 2021), the results obtained through this validation method should be taken with the grain of salt. Second, we validated our questionnaire via manual annotation of MBTI types on the answers to the open-end questions (Section 5.2).

5.1 Agreement with the MBTI Types

Given the wide popularity of the MBTI framework in non-research communities, we expected that a substantial number of AMT workers had taken the MBTI or 16Personalities tests before. In total, 340 participants responded to the optional MBTI question, with the following distribution of the MBTI traits: 87 extraversion / 253 introversion, 249 sensing / 91 intuition, 148 thinking / 192 feeling, and 206 judging / 134 perceiving.

We compared the MBTI types provided by AMT workers and the respective answers to our personality questions. To be able to compare them, we converted the 5-point scores into binary ones to match with the MBTI binary typology. For every dimension, the scores of 1 and 2 were merged as one polarity (e.g. extravert), and 4 and 5 were merged as the other polarity (e.g. introvert). The remaining 3s were considered as middle scores and were treated differently in two setups: (1) always counted as correct, regardless of the MBTI type provided by the AMT workers; (2) excluded from the analysis. In the second case, the total number of excluded cases per each dimension was: 79 (23.2%) for E/I; 76 (22.4%) for S/N; 67 (19.7%) for T/F; and 54 (15.9%) for J/P.

The middle scores either represent the cases in which participant equally exhibits characteristics of both polarities, or they indicate indecisiveness. Both types of participants could score as either of the two polarities in the MBTI questionnaires as those only offer two options. Therefore, the results of the first setup could be seen as an upper bound, and the results of the second setup a lower bound of the validity score. The exact result of the validation procedure cannot be calculated due to the limitation of the MBTI labels to capture middle cases which are common (Pittenger, 1993).

The percentage of cases in which the collected answers to our personality questions correspond to the MBTI type provided by the AMT workers is given in Table 1.

5.2 Agreement with Human Annotations

We asked two paid expert annotators, well-versioned in MBTI framework from psychology perspective, to read 30 answers to the Vacations question and 30 answers to the Hobbies questions, and annotate each of them with one of the polarities (e.g. E or I), or the label MIDDLE if they see equal amount of signs for both polarities. The annotators
were provided with the previously proposed annotation guidelines (Štajner and Yenikent, 2021) and instructed to annotate the answers to the Vacations question only for the E/I and J/P dimensions, and the answers to the Hobbies questions only for the S/N and T/F dimensions.

The 30 answers per each open-end question were randomly selected with the constraint that 10 or those answers are from workers who chose 1 or 2 as the answers to the two respective personality questions, 10 answers are from workers who chose 3 as the answers to the two respective personality questions, and 10 answers are from workers who chose 4 or 5 as the answer to the two respective personality questions.

We compared the labels obtained by human annotation (in a ternary classification task) with the labels obtained by transforming the answers to the personality questions into three classes: 1 and 2 to one polarity; 4 and 5 to the other polarity; and 3 to MIDDLE. Human annotations corresponded to the automatically obtained labels in 66.7-90.0% of the cases, depending on the annotator and the dimension in question. Several examples from this experiment are given in Table 2. Given the complexity of the task and the results of the agreement between the two trained annotators on a similar task (Štajner and Yenikent, 2021, Table 7), we find
Figure 3: Distributions of MBTI dimensions across different populations/datasets (presented in percentages of all respective users): general US population (Briggs Myers et al., 1998), Twitter users (Plank and Hovy, 2015), Reddit users (Gjurković et al., 2020), Amazon Mechanical Turk users who completed our HIT (MTURK-MBTI corresponds to those users who reported their MBTI type and is based on the MBTI type they provided, while MTURK-Q corresponds to all users and is based on the answers they chose on the scale 1–5).

6 Dataset Statistics

The total number of users whose answers we collected via Amazon Mechanical Turk platform, after the manual quality check (Section 4) was 1038.

Distributions of labels for each dimension separately, for the whole dataset (MTURK-Q) and the portion that contained MBTI types entered by AMT workers (MTURK-MBTI) is provided in Figure 3 together with the distribution of labels in different populations/datasets mentioned in Section 2. As can be observed, the distributions of labels for the S/N dimension vary significantly across different datasets, with our dataset being the only one that follows a similar distribution as the one found in the general US population. For the E/I dimension, distribution of labels in all datasets deviate from the distribution in the general US population. As already mentioned in some of the previous studies (Plank and Hovy, 2015; Verhoeven et al., 2016; Štajner and Yenikent, 2020), this is not surprising, as it is known that introverts prefer online communication (Goby, 2006). For the T/F and J/P dimensions, all datasets roughly follow the distribution in the general US population.

Figure 4: MBTI distribution (340 users).

these results satisfactory.
While being reported among the least frequent ones in the general US population (Briggs Myers et al., 1998),

Distributions of the answers to our personality questions and demographics questions are presented in Figures 5 and 6.

7 Conclusions

In this study, we proposed a set of four questions for quickly obtaining MBTI labels that better correspond to the expert human annotations of MBTI traits from short texts than the commonly used MBTI labels obtained via lengthy questionnaires. Apart from being faster to administer and allowing for obtaining large quantities of short texts on various topics, the proposed method also offers a more fine-grained MBTI typology, overcoming thus the common objections about the binary nature of MBTI questionnaires. This is particularly important for advancing research efforts in automatic MBTI personality detection from texts, as those instances that come from people that exhibit equal preferences to both polarities could be filtered out from the training data, thus lowering the noise in the models. Finally, it seems that proposed way of compiling MBTI dataset via Amazon Mechanical Turk platform leads to obtaining labels with distributions closer to those of the general US population.

Acknowledgements

We would like to thank Yassine Benajiba and Francisco Rangel for supporting this work.

References


Watching a Language Model Learning Chess

Andreas Stöckl
University of Applied Sciences Upper Austria
Digital Media Department
Hagenberg, Austria
andreas.stoeckl@fh-hagenberg.at

Abstract
We analyse how a transformer-based language model learns the rules of chess from text data of recorded games. We show how it is possible to investigate how the model capacity and the available number of training data influence the learning success of a language model with the help of chess-specific metrics. With these metrics, we show that more games used for training in the studied range offers significantly better results for the same training time. However, model size does not show such a clear influence. It is also interesting to observe that the usual evaluation metrics for language models, predictive accuracy and perplexity, give no indication of this here. Further examination of trained models reveals how they store information about board state in the activations of neuron groups, and how the overall sequence of previous moves influences the newly-generated moves.

1 Introduction
Language models are now used for a variety of applications that are not, or not directly, related to Natural Language Processing tasks, and process data that is not text Parmar et al. (2018); Huang et al. (2018); Dhariwal et al. (2020). Lu et al. (2021) used a so-called Frozen Pretrained Transformer (FPT) to study finetuning on a variety of sequence classification tasks spanning numerical computation, vision, and protein fold prediction.

In this article, we use state-of-the-art methods for language models in an area that at first glance does not seem to be an application area for them, namely the area of computer chess. This is because clear rules determine what happens here, rather than the ambiguities and vagueness that characterise language.

Brown et al. (2020) demonstrated, among other things, that a language model with increasing model capacity is able to learn the rules of arithmetic to a certain degree by training it with the data of crawled websites. In the process, elementary operations were learned in a certain number space, but not beyond. Is this limitation due to the lack of capacity of the model, insufficient training time or training data that did not contain sufficient information?

In Nogueira et al. (2021), it was demonstrated that regardless of the number of parameters and training examples, Transformer Vaswani et al. (2017) models are unable to learn addition rules that are independent of the length of the numbers seen during training.

To test the ability of language models to learn rules, and assess the influence of model size, training time and available training data, we use the commonly-stressed field of “computer chess” as an example. We investigate whether a language model is able to learn the rules of chess only from the records of games played by humans.

The test area is well suited for studying the training process of language models, since training data is available in large quantities here thanks to the recording of chess games on the booming internet chess servers. Another reason is that the quality of the language model can not only be assessed with the usual evaluation metrics for language models such as perplexity, but also with the help of the chess rules to check whether correct games are generated.

Chess as an AI testing ground has also been very popular for decades, and it regained strong focus a few years ago, thanks to the work of Deepmind1 Silver et al. (2018, 2017); Tomašev et al. (2020) together with the game of GO. There, only starting from the rules with new techniques of reinforcement learning, a chess engine was created that sur-

---

1https://www.deepmind.com/
passed everything that had previously existed in terms of playing strength.

In Schrittwieser et al. (2020), as a continuation of the previous work, knowledge of the rules is now not even assumed. When evaluating Go, Chess and Shogi, without knowledge of the rules of the game, the new algorithm MuZero achieved the superhuman performance of AlphaZero\textsuperscript{2} of the earlier work, which was trained with the rules of the game.

We will use a completely different approach as a starting point to create a system that discovers the rules of chess itself. Only transcripts of games played will be used to train language models. We will then inspect the models to ascertain how the system has learned the rules of chess.

2 Formulation of the Problem

We formulate the problem of learning to play chess within the framework of the usual methodology for language models. A language model is made capable of writing texts by training it with data comprising natural language. It achieves this by completing given passages of text, inserting word by word or part of word by part of word that is likely to be next.

In recent years, major progress has been made in this field thanks to the use of neural networks. These very powerful models are not only able to form syntactically correct sentences, but also to keep the context correct across several paragraphs, thus producing texts that are almost indistinguishable from those written by humans.

This has become possible because the new model architectures called Transformers Vaswani et al. (2017); Alammar (2018) are able to capture dependencies in the texts over long distances, and sufficient training material is also available from the WWW. In order to have a supervised training setting, the systems are either fed with text as input to guess the next word (casual language modelling), or words in a whole sentence are masked, which the model then has to reconstruct (masked language modelling). The best-known representative of the first type is the family of GPT models Radford et al. (2018, 2019); Brown et al. (2020) and for that of the second type BERT Devlin et al. (2018) and its many relatives.

We will use the GPT2 model in different model sizes as a basis and train them with chess data. This data is often in the so-called Portable Game Notation (PGN) format\textsuperscript{3}. These are text files containing some metadata, such as the names of the players, the date, the ELO rating\textsuperscript{4} and more, and the transcription of the actual game in Standard Algebraic Notation\textsuperscript{5}. This part is a string that can be seen like a sentence of a natural language. The individual moves form the “words” of the sentence.

Example:

\texttt{“d4 d5 Nf3 Nf6 e3 Bf5 Nh4 Bg6 Nxe6 hxg6 Nd2 c6 Bd3 Bd6 e4 dxe4 Nxe4 Rxh2 Ke2 Rfxh1 Qxh1...”}

Adding a new word to the \texttt{SAN} string is equivalent to making a chess move. The context that a language model has in the form of the preceding words for prediction contains all of the information needed to generate the state of the chessboard. Accordingly in principle it should be possible to create a model that predicts all legal moves of a position with positive probability and all illegal moves with probability near 0.

3 Data and Pre-Processing

For training the language model, we need a large amount of game data containing legal moves. We can download these from the internet chess server Lichess\textsuperscript{6}, for example. All games played on the server since 2013 are offered there, grouped by month. A compressed PGN file is available for each month. Overall, over 400 GB of compressed data with over 1.7 billion games played.

This is a sufficiently large amount of data, even if the amount becomes much smaller after pre-processing (e.g. removing metadata). Considering that the original GPT2 language model was trained with 40GB of internet text, we have sufficient room to experiment with different amounts of data during training.

The quality of the games played plays a subordinate role in this research, as it is initially only about learning the rules. However, it would be possible to filter games via the metadata of the ELO values of the players and examine the influence on the playing strength. However, in order to avoid games that were abandoned early on, sometimes after only one move on the server, we will use a minimum length


\textsuperscript{3}http://www.saremba.de/chessgml/standards/pgn/pgn-complete.htm

\textsuperscript{4}https://en.wikipedia.org/wiki/elo_rating_system

\textsuperscript{5}https://www.chessprogramming.org/AlgebraicChessNotation

\textsuperscript{6}https://database.lichess.org/
for filtering. The pre-processing of the batch of data is undertaken with a command line programme for manipulating the PGN files, which masters almost all of the required steps. pgnextract\footnote{https://www.cs.kent.ac.uk/people/staff/djb/pgn-extract/} can perform the required transformations in a reasonable time even with large amounts of data.

In these cleansings, all move numbers, results, comments, variations, etc. are removed from the games to obtain only the pure string with the SAN notation. One line per game is written to a file. All games with fewer than 20 moves are also filtered.

4 Related Work

A very similar approach was followed in Noever et al. (2020). With slightly different pre-processing, a model also based on GPT2 was trained from game data (11,000 games and 2.19 million games). After 30,000 training steps, plausible looking games could be generated, but with about 10 percent illegal moves in the games. It was also clearly observed that fewer errors occurred in the early stages of the game, which is obvious due to the lower context required. In our experiments, we will investigate positions after different number of moves from the starting position.

Jhamtani et al. (2018) use text generation not to generate the games themselves, but rather to generate comments in text form. A specially created data set and an LSTM (Hochreiter and Schmidhuber, 1997)-based neural network are used for training.

In our work, as in Kaplan et al. (2020); Henighan et al. (2020) we will also investigate the relationship between the development of test loss in language models as a function of model size, computing capacity and data volume. In their work, power laws were found to be observed very precisely over seven orders of magnitude, showing that the result benefits from a scale up as long as the sizes are increased simultaneously and there is no bottleneck at one. If the model sizes in terms of the number of parameters are increased by eight times, the amount of data has to be increased by eight times. We will investigate the relationships with chess-specific metrics, such as the number of correctly-generated moves.

Before we start fine-tuning the model with the games data, we would like to test whether the GPT2 models of different sizes trained with English language files are inherently capable of continuing chess games. Since the training texts comprising web crawls certainly contained chess games in algebraic notation, this could be possible.

Sequences generated with the GPT2 small, for example, if started with “e4 d5”, look like this: “e4 d5 e5 e8 f8) e5 e3 f8 e9 f8 f8) 9 0-0-0-0-0-.”. While this looks a little like a game of chess, but they are hardly correct moves.

Games generated with the medium GPT2 model look like this, for example: e4 d5 18. f6 Ne5 19. Nf3 Nd4 20. Rg1 Nd6 This looks more like chess notation, but it still does not contain correct moves. This example also shows a problem that the model has with the notation, given that games are published very differently on the web and therefore also in the GPT2 training data. They are partly with numbering and partly without, which is also reflected in the example.

The Large Model and the XL Model do not give better results. We suspect that this is due to insufficient and inconsistent lot data in the training data set. A larger model capacity does not bring any progress here.

5 Training and Evaluation of the Models

The hardware used for the training was a Kubernetes cluster Brewer (2015) with NVIDIA RTX 3090 GPUs, each with 24 GB of video RAM and 256 GB of main memory.

For the implementation, the Transformers\footnote{https://huggingface.co/transformers/} package Wolf et al. (2020) from HuggingFace\footnote{https://sgugger.github.io/the-1cycle-policy.html} based on Pytorch Paszke et al. (2019) was used.

The learning rate search was conducted according to Smith (2018). We start fine-tuning the model as the learning rate increases from very low to very high, and stop when the loss starts to truly become out of control.

Fastai Howard and Gugger (2020) was used for the training, using the 1-cycle-policy Smith and Topin (2019).\footnote{https://sgugger.github.io/the-1cycle-policy.html}

With batch sizes just fitting on the GPU memory, the models were saved after some epochs of training for the evaluations.

When language models generate sequences of words, the same sequence will always emerge if the word with the highest probability is always chosen next. Furthermore, the models often tend to repeat sequences of words. This also applies to the
generation of chess games here. Therefore, random mechanisms such as top-k sampling Fan et al. (2018) and top-p sampling Holtzman et al. (2019) are used to generate the games. These techniques reduce the tendency of repetition, although it can still occur, as Welleck et al. (2020) have investigated.

To evaluate the models, games are generated in different ways:

- From a list of typical opening positions after two moves.
- From positions of games from a game data set after a given number of moves.
- From randomly-generated positions after a given number of moves.

For all of these games, the average number of correct moves generated is counted. These three chess-specific metrics for assessing the generated moves pose different challenges to the language model. For the first evaluation criterion, it is easiest to generate legal moves, since all test positions were included in a large number of the training data games, and therefore it is sufficient for the model to remember the data. A generalization in the form that the rules of chess were actually learned is only necessary for very long generated move sequences.

The second method presents more of a challenge, increasing as the length of the given number of moves increases. Since the game data set used for the test is not included in the training data, as the length of the given moves increases, increasingly more positions will appear that the model has never seen before. Therefore, the model has to learn the rules to generate valid moves.

The third metric uses starting positions generated by a random sequence of moves. A large proportion of these moves have therefore never appeared in human games, nor in the test data set. Furthermore, the move patterns that appear are very different from those in human games, as well as from those in conventional chess programs. It is therefore very difficult for the model to generate regular moves for these sequences. Even for humans, handling such random positions is very difficult. Chase and Simon (1973) has found in experiments with chess grandmasters and amateurs that while good chess players can easily remember typical positions, they have problems with random positions.

For each trained model, the training loss, validation loss, accuracy predicting the moves in the data and perplexity are also calculated.

6 Results

For encoding, we use byte pair encoding, and therefore a typical chess game of 50 moves from both sides requires about 200 tokens for encoding the whole game. However, a game can be much longer. We use a maximum sequence length of 256 and cut of the rest of the moves.

We trained different model sizes of GPT2 (small, medium, large) with different numbers of games (99,604, 577,202, 2,163,417 games) to investigate the influence of the two factors on the learning process. To assess the results, the models were each subjected to an evaluation after a few epochs, using the evaluation metrics described in the previous section. Appendix A-1 shows how the predictive accuracy of the language model evolves with the number of GPU training hours.

The small amount of training data leads to a strong increase in accuracy for all three model sizes after only a few days of training. With more data, no model shows this increase. The different models seem to learn at about the same rate, with the small model being slightly slower.

Alternatively, if we look at perplexity as an evaluation measure, the same picture emerges. All models with a small training data set lead to a faster drop in perplexity, which indicates a better prediction of the language model. All other combinations of model size and amount of data seem to perform the same.

We now want to investigate whether the models with little data are able to learn the chess rules faster and whether it truly makes no difference with the other combinations.

With the chess-specific metrics, we can get to the root of this, and look at the performance of the models as a function of training time.

From five typical opening positions after one move by both sides, five games were generated with the models with top-p sampling (p=0.92), and then it was checked how many moves were correct until the first incorrect move was made. The average of these 25 games was calculated. Appendix A-2 shows how this performance for the respective models developed with the training time.

Top-p sampling is used to check the models’ ability to produce not only the most likely move,
but also other valid moves. In a chess position, more than one move is usually possible.

For each combination of models and data set, the result was plotted over the training time and a logarithmic fit was drawn. No data are available for the GPT2 large model with the larger amount of data because hardly any models could be trained in the available time period due to the long training time per epoch.

In contrast to the accuracy, this measure shows that the models with a small amount of data perform significantly worse.

The best values are delivered by the medium model with medium data volume. This is in sync with the observation in Kaplan et al. (2020) that model size and data volume should be increased together for good results.

The second metric is a more demanding task, as the games are not generated from a typical starting position after one move, but rather from positions after ten moves, taken at random from games played. These games were not from the training data set.

Games are generated from 100 random positions using top-p sampling (p=0.92), and then tested to see how many moves were correct from this position, whereby the mean value is then calculated. Appendix A-3 shows how this performance for the respective models developed with the training time. Again, a logarithmic fit was drawn.

For this task, all models benefit from more training data, although there are no differences in model size.

As a third metric, we have chosen a task that is even more challenging, and it should make it clearer whether the task was solved based on learned rules rather than pure memorisation of variants.

Again, starting with top-p sampling (p=0.92), 100 games were generated from one position after ten random moves, and then checked to see how many moves were correct from this position, whereby the mean value was then calculated. Appendix A-4 again shows how this performance for the respective models developed with the training time.

It can be seen that this task is much more difficult, as the models now only manage to correctly execute sequences of a few moves on average. However, the same effect can be seen as with the two previous metrics, namely that all models benefit from a larger amount of data. In terms of model size, there are few differences.

7 How Is Chess Knowledge Stored in the Model?

So far, we have explored how language models benefit from more parameters in the model and more training data when learning chess rules. Now we want to ascertain whether there are patterns in how the information of the chess rules is stored in the parameters of the models. We will use different visualisations for this purpose.

For this purpose, different approaches help to visualise language models. On the one hand, we look at the influence of different inputs on the generated words/moves, as shown in Arrieta et al. (2020); Li et al. (2015), and on the other hand, we can look at the activation of the different neurons in the models, as shown in Karpathy et al. (2015); Poerner et al. (2018); Dalvi et al. (2019).

When properly visualised and studied, neuron activations can reveal the roles played by individual neurons and groups of neurons. We use the Eccox library10 Alammar (2021) for analysis.

In order to combine the groups of neurons involved in the same tasks, factorisation methods for matrices are used, whereby the library used employs NMF for this purpose.

Let us look at the inner workings of a trained model for moves from some sample positions. The first position is from the opening phase and it is a special situation where only one legal move is possible (Fig. 1).

If we look at the influence of the individual parts of the sequence in Fig. 2, the moves that led to position Fig. 1, on the new move, we see that the last parts have the strongest influence, but otherwise the entire sequence also influences the output.

The colour code shows the strength of the influence, and alternatively we can also show the influence of the individual parts as a percentage (Fig. 3).

How certain is the model that the generated move is a correct one?

For this purpose, we look at the probabilities that the model assigns to the individual possible tokens at the end of the last layer of the generator part. The move is generated in two parts, and it shows a high probability of the generated move for both

10https://www.eccox.io/
Let us now look at the activations of the neurons and try to see how the model stores information about the chessboard from the move sequences. In order to generate valid moves, the language model needs a representation of the chessboard and its pieces in the states of the neurons.

If we look at the activation of the neurons by group, we see that one group (red in Fig. 5) is active on the parts of the text responsible for the row information on the chessboard, and another for the column and piece information (blue in Fig. 5). The other two are active at the beginning or end of the sequence.

Let us now look at another position, which comes from the so-called “Game of the Century”\(^1\). In the complicated position from the middle game in Fig. 6 with many possible moves, the model reaches its limits. It recognises that a piece on the d-file was captured last and wants to capture it back with Rxd or Qxd. However, there is no valid move for this in this position. The representation on the board was not correctly mapped in the neurons here, so no valid moves are generated.

The probability for possible next tokens in the last layer of the network is also distributed over

---

\(^1\)https://en.wikipedia.org/wiki/The_Game_of_the_Century_(chess)
many candidates (Fig. 7).

As in position 1, virtually the entire sequence of previous moves has an influence on the new move to be generated, as can be seen in Fig. 8.

In this example, again the specialisation of groups of neurons on the row information and the figure and column information can be seen. However, the activations are lower at the beginning of the sequence.

8 Conclusion and Future Work

In the two example positions, it can be seen that the whole sequence has an influence on the generated move, which is necessary to generate correct moves.

By looking at the activation of neurons, we could see that the information about the row, column and type of figure is stored in different groups of neurons. Thus, the model seems to organise the storage of the information necessary to represent the state of the board. So the training could benefit from a longer string representation of the games such as the long algebraic notation since the mechanics of moves are more directly accessible. Because rows, columns, and pieces are all represented as separate tokens.

In order to study the learning process for larger models and larger data sets we would like to use distributed training with Falcon (2019) with higher computational capacity, and the optimizations proposed in Rajbhandari et al. (2020); Rasley et al. (2020); Zhang and He (2020); Ren et al. (2021); Tang et al. (2021); Rajbhandari et al. (2021); Li et al. (2021).

We have only investigated the learning of the rules of the game here. Accordingly, investigating the possible playing strength of the language models as a function of the training data quality would be an interesting extension of the investigations. The influence of the selection of the training data according to the playing strength of the players involved and the model size would have to be considered here.

References


Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 3505–3506.


Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam
Appendix A-1: Predictive accuracy over training time

Appendix A-2: Average number of correct moves from an opening position
Appendix A-3: Average number of correct moves from positions after move 10 from games played

Appendix A-4: Average number of correct moves from position after ten random moves
Tackling Multilinguality and Internationality in Fake News

Andrey Tagarev
Research Unit, Sirma AI
79 Nikola Gabrovski St
Sofia, Bulgaria
andrey.tagarev@ontotext.com

Ivelina Nikolova-Koleva
Bulgarian Academy of Sciences and Sirma AI
79 Nikola Gabrovski St
Sofia, Bulgaria
ivelina.nikolova@ontotext.com

Krasimira Bozhanova
Semantic Analytics Solutions, Sirma AI
79 Nikola Gabrovski St
Sofia, Bulgaria
krasimira.bozhanova@ontotext.com

Ivan Ivanov
Life Science Solutions, Sirma AI
79 Nikola Gabrovski St
Sofia, Bulgaria
ivan.b.ivanov@ontotext.com

Abstract

The last several years have seen a massive increase in the quantity and influence of disinformation being spread online. Various approaches have been developed to target the process at different stages from identifying sources to tracking distribution in social media to providing follow up debunks to people who have encountered the disinformation.

One common conclusion in each of these approaches is that disinformation is too nuanced and subjective a topic for fully automated solutions to work but the quantity of data to process and cross-reference is too high for humans to handle unassisted. Ultimately, the problem calls for a hybrid approach of human experts with technological assistance.

In this paper we will demonstrate the application of certain state-of-the-art NLP techniques in assisting expert debunkers and fact checkers as well as the role of these NLP algorithms within a more holistic approach to analyzing and countering the spread of disinformation. We will present a multilingual corpus of disinformation and debunks which contains text, concept tags, images and videos as well as various methods for searching and leveraging the content.

1 Introduction

The topic of fake news and intentional spread of disinformation has been gaining increasing prominence over the last decade. The distinction of terminology and attempts to classify various erroneous or misleading statements online is constantly evolving but disinformation, as shown in Fallis (2015), has recently settled as the commonly accepted term to describe the intentional and systematic spread of incorrect information.

The spread of this incorrect information is strongly reliant on social media, causing a strong emotional reaction and quick propagation before its inaccuracy can be effectively exposed. This means that in most cases minimal effort is put into crafting the disinformation, instead relying on speed, volume and reuse of slightly modified pre-existing materials. There are, of course, always new disinformation materials popping up but they are in the minority and should they gain traction, they will very quickly get picked up, modified slightly and reused.

A very stark example of this kind of interaction was provided in the early months of the Covid pandemic as reliable scientifically-tested information was still rather scarce and the void was filled by a wide variety of rapidly-spreading fake and unsupported claims. This can be viewed both from the perspective of journalists mobilizing to counteract the spread and from that of researchers looking into assisting their efforts e.g. in identifying spread of disinformation that has already been debunked as in Singh et al. (2021) or in tracking the comparative effect of disinformation and debunking tweets as in Jiang et al. (2021).

This means that combating the spread of disinformation can happen on a variety of levels. One option is to identify the creator and limit their reach - a replacement will pop up eventually but rebuilding a presence in the social network requires time and resources. Alternatively, it is possible to identify a piece of disinformation early in its spread and expose it to the people who interact with it before it really gains traction. Finally, it is possible to monitor social media for trending topics and work on creating convincing well-supported debunks to new disinformation that has gained popularity. An ideal approach would combine all three aspects in some manner.

Finally, the reality is that while disinformation might typically involve minor or simple modifi-

1https://weverify.eu/blog/speeding-up-the-debunking-process/
cations, it is still intentionally crafted to be misleading and is constantly evolving and adapting. This makes completely automated approaches to combating it impractical. Meanwhile, the sheer volume of information that needs to be tracked, analysed and correlated makes a completely manual approach equally impractical. The solution will inevitably then involve a hybrid approach.

To that end, we present a data set based on a collection of fact-checker created debunks of pieces of disinformation that has been extended with additional metadata. Several forms of advanced search functionality have been developed on top of it in order to make discovering relevant content in the data set as straight-forward as possible. This will allow fact-checking experts to easily check for previous work on disinformation they encounter, point to previous instances of it being used and react quickly to its spread in order to counter it early on.

2 The Data Set

The data set used for our experiments is based on a snapshot of the Database of Known Fakes (DBKF). It is a collection of debunking content from highly respected fact-checking organizations around the world extended with additional metadata related to said debunks in order to enable the advanced search and correlation functionality we present here.

Figure 1 shows an overview of the major types of objects contained within the DBKF and the connections between them. At its core, the data model of the data set is based around the Claim and Claim-Review format defined within schema.org which is already familiar to and used by many fact-checking organizations.

The core objects defined within the schema.org specification are a Claim (a short statement summarizing the target of the debunk) and ClaimReview (a typically article-length debunk of the claim being discussed). As can be seen in Figure 1, these two objects are extended with additional explicit objects. The two most important additions areAppearances and Evidences - the former are links to posts where a specific Claim is being made and the latter are external content supporting the explanation and reasoning contained within a debunk.

Appearances are automatically expanded to include additional metadata available at the external website (more on that in Subsection 3.1) and a number of state-of-the-art systems are used to enrich the objects further (more on that in Subsections 3.3 and 4.2). Some of this metadata is more detailed and contained within specialized objects such as Image, Video, Concept and Annotation (which is an instance of a concept at a specific location in a document’s text).

At the time of writing, the data set contains

- **Claims**: 32,138
- **Debunks**: 32,220
- **Appearances**: 74,099
- **Evidences**: 348,100
- **Concepts**: 110,158
- **Annotations**: 359,032
- **Images**: 9,774 links to an image of which 9,217 unique image urls
- **Videos**: 9,866 links to a video of which 9,745 unique video urls

but the numbers are always growing as new input is being collected from many fact checking and debunking organizations daily.

Figure 2 provides a specific example of a debunk that might be retrieved by the system (pre-advanced enrichment steps). The claim is that an immigrant destroyed a statue in Italy, three appearances provide links to three tweets that made that claim, the debunk disproves the claim and explains what actually happened while the evidences support that explanation.

2.1 Sources

The contents of the data set are sourced from organizations in the Facebook third-party fact-checking program. At the time of writing the data set contained data from 17 separate organizations in at least 13 languages and based on disinformation encountered in over 20 different countries.

Due to the variety in article formatting and institutional approaches to writing the debunks, there is some significant variety in the details of the retrieved debunks but the objects in the high level model presented in Section 2 are present for all

---

2. https://schema.org/Claim
simplification of existing debunking content and serve as a comprehensive repository.

3 Enrichment

This section will go over the additional enrichment steps that are used to take the DBKF beyond a simple collection of existing debunks and unlock the ability to do advanced searching and correlation by expanding the metadata available on the debunks. This contains the additional metadata added to Appearance and Evidence objects, language tagging of all texts and named entity recognition.

3.1Appearances and Evidence

When originally extracted from the debunking articles, appearances and evidences are just plain urls. These objects are expanded to include additional metadata, whenever possible. This has multiple goals:

1. Extract text, images, videos, author information, publication time, etc. to be used in searching, filtering and analysis

2. Archive the link so that it is still accessible should the original be removed (often the case with disinformation that gets debunked)

3. If the link is already an archive, extract the original url for domain analysis purposes

The retrieval of this metadata is, of course, not always possible. Aside from the common case of a link being removed or simply inaccessible, there is also no guarantee how the target website will be formatted. To that end we have chosen...
to focus metadata expansion on a few social media websites that are particularly common (Twitter, Facebook, YouTube), popular archive websites (perma.cc, archive.org, archive.is, etc.) and websites that follow the Google guidelines for publishing news articles with properly tagged metadata. This allows us to collect at least some metadata for over 90% of appearances.

3.2 Language Recognition

Language detection is a simple task but an important first building block for some of the more complex enrichment steps presented later such as for selecting the appropriate NER pipeline in subsection 3.3. For this reason we ran the texts of claim, debunks and appearances through a well-tested language detection algorithm- Shuyo (2010).

It is worth noting that while language detection is not an especially difficult task, the contents of the data set are quite varied in a number of ways - quite a variety of languages, mixed-language texts, various lengths (from few words to multi-page articles). This all means that some amount of errors will inevitably be introduced at this step of the process.

In Figure 3 we can see the language distribution produced by the algorithm. The distribution of languages corresponds to what we expect to see based on the fact-checking sources present in the data set. English, Spanish, German and French are currently the major languages and there is a long tail of languages where the total number of available debunks is much lower.

It is worth keeping in mind that the distribution is both a reflection of currently active fact-checking organizations and the specific sources that are processed and ingested in the system. This is to say that the situation is quite fluid and more languages can become relevant in the future.

3.3 Locations and Concepts

The language tagging of all text in the data set allows the final step in metadata enrichment - named entity recognition carried out over the corpus. This task was further divided in two, based on the needs of the users and analysis of available algorithms. After reviewing the literature on comparative analysis of available algorithms Schmitt et al (2019), we ran some additional comparisons of different approaches since the diverse nature of our data set and unconventional target concepts make comparison over standard data sets less feasible.

Table 1 shows a comparison between an in-house developed NER pipeline targeted at the publishing domain, which is based on GATE\(^6\) using a subset of the Wikidata\(^7\) data set (referred to as CES - Concept Extraction Service), the default spaCy pipeline\(^8\) and the NER functionality of Google Cloud\(^9\). The comparison was carried out over a variety of language although the CES algorithm was only used on English texts.

<table>
<thead>
<tr>
<th></th>
<th>Detected</th>
<th>P (strict)</th>
<th>P (partial)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CES</td>
<td>156</td>
<td>0.74</td>
<td>0.95</td>
</tr>
<tr>
<td>spaCy</td>
<td>171</td>
<td>0.66</td>
<td>0.83</td>
</tr>
<tr>
<td>Google Cloud</td>
<td>190</td>
<td>0.59</td>
<td>0.81</td>
</tr>
</tbody>
</table>

The conclusion is that CES, while limited to only English, has a notable performance advantage. Meanwhile the spaCy pipelines and Google Cloud offering are roughly on par but both support a larger variety of languages.

For that reason, CES was used to do NER of location mentions over the data set. It was decided that the better performance in correctly identifying location mentions offsets the limiting of that enrichment to only English. Conversely, the general "concept" tags rather typical POL entities are more useful when applied to as many of the texts as practical.

A word of caution on the applicability of simple numerical comparison in the case of general con-

\(^{6}\)https://gate.ac.uk/
\(^{7}\)https://www.wikidata.org/
\(^{8}\)https://spacy.io/usage/models/
\(^{9}\)https://cloud.google.com/natural-language
cept recognition. The case of disinformation spread is a very challenging one for the NER task since the uniquely identifying concepts associated with given misinformation are quite varied and much less well-defined than the typical POL formulation of the task. Words like "immigrant", "hospital", "lemon", "vaccine", "5G" and many others have actually proven quite important but their popularity and usefulness is actually quite limited in time. To that end we placed particular importance on the system's ability to identify such concepts when they first begin to gain prominence.

4 Search

The final step in unlocking the full potential of the data set is to enable powerful search functionality that can make discovering existing debunk information and locating similar cases of disinformation in the past as quick and easy as possible. This kind of searching should utilize the full capability of the collected and enriched data set and can be used as a stepping stone to semi-automated and automated systems such as early detection of disinformation and chat bots. It is also a first step to being able to detect larger trends within the data such as tracking the spread of a particular disinformation claim across countries or watching a particular piece of disinformation change and evolve in response to fact-checker debunkers.

There are several aspects of this search functionality that build on each other. Firstly, we will describe the full-text and faceted search then near-duplicate detection based on visual similarity and multilingual search that uses latest neural machine translation. Subsection 5.1 will also briefly discuss what we envision these search capabilities building to in the future.

4.1 Facets

The basic functionality needed by expert fact-checkers is the ability to quickly look for debunkers related to the claim they are currently investigating using keywords or even phrases. To this end, we have implemented a full-text search based on the Elasticsearch engine. Results from the full-text search are presented, based on the user preference, either by the relevance score returned by Elasticsearch, or by date of publication. In order to fully exploit the information collected from the sources, as well as the metadata and enrichments created while populating DBKF, we have implemented faceted search. The facets currently support filtering and slicing DBKF content or search results by: language, author, debunk publisher (source), time of publication, locations and concepts. For example, using the full-text search with the "5G" keyword, together with facets can help debunkers quickly find the false claims that were circulating in different languages on the subject during a specified timeframe. Similarly, fact-checkers can use facets to quickly check what locations are mentioned in "vaccine" (selected from the concept facet) related disinformation. The ability to search DBKF with the help of facets can be beneficial not only to verification professionals but also to researchers in the field of disinformation, social scientists and policy makers.

4.2 Visual Similarity

Searching based on visual similarity relies on the research performed by Kordopatis-Zilos et al. (2019) and is carried out by using the near duplicate detection (NDD) service. This service supports indexing and searching for both images and videos based on visual similarity between the contents. This means that every image or video discovered within appearance and evidence objects is automatically indexed within the NDD service in order to be available for visual similarity searches.

When a visual search is initiated by the user, the image or video they provide is also indexed into the NDD service and then all returned results are tied back to their corresponding debunkers within the DBKF. This effectively enables us to discover debunkers that contain similar images and videos even if they have been reuploaded or slightly modified which are the typical way bad faith actors reuse them for spreading disinformation.

It is worth noting that once indexed, an image or video does not need to be stored in its original form and, in fact, due to concerns about storage and distribution rights of digital content, they are usually not stored locally. Instead, the final representation of the visual object is a single vector in a highly-dimensional space which cannot be used to recreate the original digital object. In practice, this means that the visual similarity service can discover connections to similar content but cannot show that content to the user.

As a practical step to combat the frequent disap-
pearance of content tied to disinformation, we work with internet archiving websites to preserve any Appearance when we initially encounter it. These websites have a procedure for the content owner to have the a specific archived item removed but in practice while the social media post often disappears within months, the archiving organizations are rarely contacted to have the archived content removed. So as a response to a visual similarity search, we provide the similar item, a link to its original url and a url to an archiving website making it quite likely but not guaranteed that the user can view the original image or video.

There are other image and video processing services available for integration with the contents of the DBKF such as automated image forensic analysis Zampoglou et al. (2016) and deep fake detection Charitidis et al. (2020) but those are more suited to producing evidence to support debunks than for searching the data set. That said, it is possible to extend the metadata associated with debunks to reflect the kind of visual manipulations encountered within a piece of disinformation and make that available for search as another facet similar to the ones described in Subsection 4.1.

4.3 Multilingual

The newest kind of search functionality enabled in the DBKF focuses on the multilingual aspect of the data set. Its intent is to vastly improve the ability to track the spread of disinformation in international situations.

The search utilizes the latest advances in neural machine translation and the translation is based on M2M-100 - a many-to-many multilingual translation model presented in Fan et al. (2020). It supports bidirectional translation between any pair of over 100 languages and shows a marked improvement in translation between non-English languages when compared to English-centric model. English is often not the first language in which disinformation appears so this is a very useful feature for our use case.

As shown in Figure 4, we have chosen to focus on the eight major languages of the data set. This is a reflection of the analysis shown in Figure 3 about the current distribution of data in the data set. A major advantage of the M2M-100 model is that it allows seamless adaptation to changes in the available data.

To quickly summarize the workflow presented, the search first identifies the language of the query. If it is a supported language, it translates it into all other supported languages and sends off a multi-expression query to be processed. Otherwise it forwards it without modifications. The search returns the results in order of relevance without regard to which translation they have matched.

The decision to not translate queries in unsupported language is a reflection of the limitation of the search. If the search query is too short or ambiguous (a not unlikely situation), the language tag will be unreliable and the translations will likely be of equally low quality.

5 Conclusion and Future Work

In conclusion, the DBKF already contains a large amount of data extended with useful metadata and powerful search capability. This can make it a powerful tool in the arsenal of fact-checkers and also allows its incorporation in counter-disinformation campaigns where people are targeted with evidence of a claim’s falsehood before they spread it unknowingly. The contents of the database are also constantly growing with the automatic ingestion of new content. Future developments can include.
the addition of new fact-checking sources, support for metadata-expansion of more types of social media posts and further building on the modeling and search functionality.

5.1 Multimodal Search

One improvement of particular interest is the option to enable a true multimodal search over the data set. As discussed in Section 4, we already have full-text, faceted, image and video search so the next step would be to combine them into a single endpoint. This would enable to effortlessly search for debunks relevant to a social media post, essentially the automatic ability to ask "Is this post repeating known disinformation?"

The challenge is actually combining the various results in a meaningful way. The various modalities operate on completely different scales, not to mention that they are all optional and possibly multi-valued e.g. how do you compare a post with a sentence of text, three concepts, a location mention and five images to a debunk that has three pages of text, seventy concepts, no images and two videos? It is by no means an insurmountable obstacle but extensive experimentation and careful fine-tuning will be required to produce intuitive and helpful results.

5.2 Model Extension

There are various ideas for adding additional aspects to the data model. One idea briefly mention in Subsection 4.2 is tagging debunks based on the type of disinformation techniques they represent e.g. deep fake videos, out-of-context images, etc. Work has began on building a vocabulary for disinformation techniques but we are consulting with fact-checking experts to align it to their expectations and needs.

The more interesting but complex direction of expansion would be to incorporate deeper understanding and tracking of disinformation campaigns into the model. This would allow to explicitly connect individual debunks into the larger trends they are coming up against.

Acknowledgments

This work is funded by the EU H2020 WeVerify (grant agreement: 825297) project.

References

Polychronis Charitidis, Giorgos Kordopatis-Zilos, Symeon Papadopoulos, and Ioannis Kompatsiaris. 2020. Investigating the impact of pre-processing and prediction aggregation on the deepfake detection task.


Nakatani Shuyo. 2010. Language detection library for java.


Learning and Evaluating Chinese Idiom Embeddings

Minghuan Tan and Jing Jiang
School of Computing and Information Systems
Singapore Management University
mhtan.2017@phdcs.smu.edu.sg jingjiang@smu.edu.sg

Abstract

We study the task of learning and evaluating Chinese idiom embeddings. We first construct a new evaluation dataset that contains idiom synonyms and antonyms. Observing that existing Chinese word embedding methods may not be suitable for learning idiom embeddings, we further present a BERT-based method that directly learns embedding vectors for individual idioms. We empirically compare representative existing methods and our method. We find that our method substantially outperforms existing methods on the evaluation dataset we have constructed.

1 Introduction

Chengyu (成语, literally meaning “set phrases”) are a type of idiomatic expressions in Chinese that usually consist of four Chinese characters. They are mostly derived from ancient Chinese literature and many of them are based on historical stories. The semantic meanings of Chengyu are often non-compositional and sometimes metaphoric. For example, the Chengyu 瓜田李下 literally means “melon field, beneath the plums,” but its idiomatic meaning is to warn people to avoid situations where a person may be easily suspected of wrongdoing. Chengyu are commonly used in modern Chinese language, and using computational methods to understand Chengyu plays an important role in Chinese language understanding. For example, a recent work studied how to improve essay writing with recommending Chinese idioms (Liu et al., 2019), and others studied how to improve reading comprehension by correcting usage of Chinese idioms (Wang et al., 2020) and differentiating synonyms of Chinese idioms (Long et al., 2020). In this paper, we refer to Chengyu as Chinese idioms, although there are also other types of idioms in Chinese.

Recent years have witnessed the success of deep neural networks for many NLP tasks. A central idea behind deep neural networks for NLP is to use dense embedding vectors to represent language units including words, phrases and sentences, and such embeddings have been shown to be useful for many tasks such as sentiment analysis (Yu et al., 2017), question answering (Hao et al., 2017) and machine translation (Zhou et al., 2016). We therefore believe that it is also desirable to derive embedding vectors for Chinese idioms that can accurately capture their semantic meanings. However, it is not clear whether existing methods for Chinese word embeddings are effective in deriving good Chinese idiom embeddings, and there are at least two reasons for this.

First, existing Chinese word embedding evaluation datasets do not have sufficient coverage of idioms. For example, in the commonly used WordSim-240 (Wang et al., 2011) and WordSim-296 (Chen et al., 2015) datasets for Chinese word relatedness, no idiom is found. More recently, Huang et al. (2019) released a COS960 dataset with similarities of Multiword Expressions (MWEs). Although COS960 covers 150 Chinese idioms, this is still a relatively small number, and only 20 MWE pairs in COS960 consist of both idioms. For the word analogy task, another commonly used evaluation task, Chen et al. (2015) created the first Chinese dataset with 1,125 analogies, but no idiom is included. Li et al. (2018) released a large and balanced dataset CA8 for word analogy. Although CA8 has 400 entries that contain idioms, this is still a relatively small number, and only 20 MWE pairs in COS960 consist of both idioms. For the word analogy task, another commonly used evaluation task, Chen et al. (2015) created the first Chinese dataset with 1,125 analogies, but no idiom is included. Li et al. (2018) released a large and balanced dataset CA8 for word analogy. Although CA8 has 400 entries that contain idioms, they only cover 32 unique idioms and no idiom pairs are included. With this lack of coverage of idioms in existing evaluation datasets, we cannot judge whether existing Chinese word embedding methods work well for Chinese idioms.

Second, it is reasonable to suspect that existing word embedding methods for Chinese have
limitations that make them less suitable for Chinese idioms. For non-contextualized word embedding methods such as Continuous-Bag-Of-Words (CBOW) and Skip-Gram with Negative Sampling (SGNS), they treat contexts as bags of words, but given the complex meanings of Chinese idioms, learning their embeddings from bag-of-word representations of contextual words without considering the order and interactions between these contextual words may not be sufficient. Existing pre-trained non-contextualized Chinese word embeddings are also usually trained with a relatively small context window, but the semantic meaning of a Chinese idiom is often based on a larger context where the idiom appears. In fact, it has been observed that larger context windows result in more topicality (Levy and Goldberg, 2014; Bansal et al., 2014), and we suspect that for learning Chinese idiom embeddings a larger context window helps. Therefore, existing pre-trained non-contextualized Chinese word embeddings may not capture the semantic meanings of Chinese idioms well. On the other hand, recent contextualized word embedding methods such as BERT (Devlin et al., 2019) and its variants (e.g., ERNIE (Zhang et al., 2019)) consider longer contexts and use attention mechanism to model interactions between words, but since they do not focus on learning word embeddings, they do not learn a single embedding vector for each Chinese idiom. Although we can aggregate the character-level representations of the characters inside an idiom and treat the aggregated representation as the idiom embedding, since many Chinese idioms' semantics are non-compositional, this simplified approach is likely not ideal.

In this paper, we study the problem of learning and evaluating Chinese idiom embeddings. To overcome the first challenge stated above, i.e., the lack of suitable evaluation dataset for Chinese idiom embeddings, we construct an evaluation dataset that contains Chinese idiom synonyms and antonyms. We also define two evaluation metrics to measure how close the ground truth idiom synonyms are in an embedding space in order to quantify the quality of the embedding space. To overcome the second challenge stated above, i.e., the potential limitations of existing word embedding methods for Chinese idioms, we propose to adapt a method (Tan and Jiang, 2020) for Chinese idiom recommendation to learn idiom embeddings. This method learns a single embedding vector directly for each idiom and encodes the contextual information using BERT.

With the evaluation dataset we have created, we empirically compare a SGNS-based non-contextualized word embedding method for Chinese, two variants of BERT for Chinese, and our Chinese idiom embedding method. We find that based on the two metrics we have defined to measure closeness of synonyms in an embedding space, our method performs substantially better than existing methods. We also find that our method can better distinguish idiom antonyms from idiom synonyms than existing embedding methods. We also conduct further analysis to demonstrate that embedding methods that rely more on Chinese character information show advantages only when the synonyms share many common characters.

The contributions of our work are twofold: (1) We construct an evaluation dataset to facilitate the evaluation of Chinese idiom embeddings. Code and data are released on github. (2) We present a BERT-based method that directly learns Chinese idiom embeddings, and we empirically compare this method with existing Chinese word embedding methods to demonstrate both the importance of learning a single embedding vector for an entire idiom and the importance of using BERT to encode the context when learning these idiom embeddings.

2 Related Work

Word Embeddings

Word embedding is an important technique in NLP. It computes dense meaning representations for discrete words. It is built upon the distributional hypothesis that linguistic items with similar distributions have similar meanings. Several methods have been proposed to learn non-contextualized word embeddings efficiently, including Continuous Bag-Of-Words (CBOW), Skip-Gram with Negative Sampling (SGNS) and GloVe (Pennington et al., 2014). In this paper, we use an SGNS-based Chinese word embedding method as a representative non-contextualized word embedding method for evaluation. Contextualized word embeddings such as ELMO, GPT and BERT have been developed in recent years and shown their high effectiveness for many NLP tasks. In this paper, we use two representative BERT variants, BERT-wwm and ERNIE, to evaluate Chinese idiom embeddings derived from pre-trained Chi-
Chinese BERT models.

**Evaluation of Chinese Word Embeddings** For word embeddings, existing evaluation methods can be categorized into intrinsic and extrinsic methods (Schnabel et al., 2015). Commonly used intrinsic methods include word similarity and word analogy, while extrinsic methods rely on downstream NLP tasks (Pennington et al., 2014). In this paper, we use an intrinsic method to evaluate Chinese idiom embeddings.

Several benchmark datasets for evaluating Chinese word embeddings have been released (Wang et al., 2011; Finkelstein et al., 2001; Jin and Wu, 2012; Chen et al., 2015; Guo et al., 2014; Huang et al., 2019; Li et al., 2018). But as we pointed out in Section 1, existing datasets have low coverage of Chinese idioms.

**Neural Network Models for Chinese Idiom Understanding** Despite the importance of Chengyu in Chinese language understanding, there have been only a few pieces of work on Chengyu using neural models (Jiang et al., 2018; Liu et al., 2019; Zheng et al., 2019). Chinese Chengyu Recommendation (CCR) has been addressed in recent years (Liu et al., 2019; Jiang et al., 2018; Zheng et al., 2019). In this paper, we adapt a method for CCR (Tan and Jiang, 2020) to learn Chinese idiom embeddings.

### 3 Construction of the Evaluation Dataset

A standard intrinsic task for evaluating word embeddings is word similarity (Bakarov, 2018; Wang et al., 2019). For Chinese idioms, a natural choice of idiom pairs that are semantically similar are synonyms or near-synonyms. Although previously Wang et al. (2013) constructed a Chinese idiom knowledge base that contains idiom synonyms, this knowledge base is not publicly available. On the other hand, there exist online resources containing synonyms and near-synonyms of Chinese idioms. We choose two websites, kxue.com (快学网) and Baidu Baike (百度百科), as the sources from which to crawl idiom synonyms and near-synonyms. We also collect idiom antonyms from these two websites because an antonym of an idiom is often topically related to that idiom and therefore may be also close to that idiom in an embedding space. However, we expect a good idiom embedding method to be able to separate antonyms from synonyms.

**Idiom Vocabulary:** According to Wang et al. (2013), there are in total around 38k Chinese idioms, among which around 3.5k are commonly used. In order to obtain a vocabulary of Chinese idioms with high coverage, we merge the idioms found in the following four resources: (1) Chengyu Daquan5, (2) Xinhua Chengyu Dictionary6, (3) Chengyu Cloze Test7, and (4) ChID.8 This gives us a Chinese idiom vocabulary with 33,237 idioms.

**ChIDsyn:** As we have pointed out earlier, we believe idiom synonyms can help us evaluate idiom embeddings. To construct a large dataset of Chinese idiom synonyms, we crawled synonyms from two websites: (1) Kxue.com is an online Chinese thesaurus. It has a dedicated page where Chinese idiom synonyms are listed. Each entry in this list consists of a key and a value, where the key is a Chinese idiom and the value is one or more other Chinese idioms that are near-synonyms of the key. We crawled all the entries from this idiom synonym page on kxue.com.8 Baidu Baike is an online encyclopedia in Chinese. For each idiom, there is a section called 成语辨析 (Chengyu Differentiation) that lists its synonyms and antonyms.9 We crawled the synonyms of those idioms in our vocabulary that can be found on Baidu Baike. In total, we obtained around 30k entries of Chinese synonyms. We then removed those idioms in the data that are not in our idiom vocabulary as described earlier. In the end we obtained a total of around 21K entries in our synonym dataset, where each entry consists of a query idiom and a set of other idioms that are the query idiom’s synonyms or near-synonyms.

We observe that a significant portion of the synonyms share common characters with the query idioms. For example, 山盟海誓 (oath of eternal love) and 海誓山盟 are treated as near-synonyms.

---

5<https://github.com/zhengcj1/ChID-Dataset>
7<https://github.com/bazingagin/chengyu_data>
8<https://github.com/zhengcj1/ChID-Dataset>
in our dataset, but these two idioms contain exactly the same set of Chinese characters. In fact, they are variants of the same basic form. Another example is 挨家挨户 (door to door) and 挨门挨户, which share three common characters. In general, it is not uncommon for Chinese idioms to have such variants due to historical reasons such as misuse (including literary malapropism). Although these are valid near-synonyms, we suspect that they may affect the evaluation of idiom embeddings. This is because those idiom embeddings that rely more on character-level information are likely to gain advantages when evaluated on these near-synonym pairs sharing common characters. For example, if an idiom embedding is obtained by averaging the character embeddings of its component characters, then it is very easy for this type of idiom embeddings to recognize that 山盟海誓 and 海誓山盟 are near-synonyms (because they would have the same average character embedding), but we would not be able to know whether such embeddings truly capture the semantic meanings. We also suspect that for those idioms that have near-synonyms sharing common characters, their semantic meanings are more likely to be compositional and thus less idiomatic. For example, for the idiom 挨挨挨户, the character 挨 means “in sequence” and both 家 and 户 mean “household.” The meaning of the idiom, which is “door to door,” can be directly inferred from the meanings of the characters. Therefore, when the character 家 (household) is replaced with the character 门 (door), the meaning of the idiom remains the same.

Consequently, we move those synonyms that share at least two common characters with the query idioms into a separate dataset, which we will not use as the main evaluation dataset. The remaining synonyms always have no more than one common character with their query idioms. We refer to this cleaned synonym dataset as ChIdSyn, and the separate dataset containing synonyms sharing two or more common characters is referred to as ChIdSyn-com. We will use ChIdSyn-com for additional analysis in our experiments. Statistics of ChIdSyn and ChIdSyn-com can be found in Table 1.

**ChIdAnt:** From the same two websites, we have also collected around 10K entries in an antonym dataset which we refer to as ChIdAnt. Similarly, each entry in this dataset consists of a query idiom and its antonyms. Although antonyms are idioms having opposite meanings, they are often topically closely related. For example, the idiom 饱学之士 means “a scholarly man,” and its antonym 胸无点墨 means “uneducated.” We can see that their meanings are topically closely related. We therefore suspect that they are still close in an embedding space, but ideally a good idiom embedding method should be able to distinguish the synonyms of a query idiom from its antonyms. Table 1 gives some statistics of ChIdAnt.

### Table 1: Statistics of the crawled datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Before Filtering</th>
<th>After Filtering</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Idioms #Entries</td>
<td>#Idioms #Entries</td>
</tr>
<tr>
<td>Crawled</td>
<td>33,524 30,354</td>
<td>21,745 20,753</td>
</tr>
<tr>
<td>ChIdSyn</td>
<td>11,387 8,897</td>
<td>8,125 6,822</td>
</tr>
<tr>
<td>ChIdSyn-com</td>
<td>28,622 24,147</td>
<td>18,498 15,836</td>
</tr>
<tr>
<td>ChIdAnt</td>
<td>11,263 9,733</td>
<td>7,939 7,316</td>
</tr>
</tbody>
</table>

Existing Chinese word embedding methods can be used to derive idiom embeddings. However, as we have discussed in Section 1, they may not be ideal for learning Chinese idiom embeddings. In this section, we first briefly review existing Chinese word embedding methods and how we use them to obtain idiom embeddings. We then present a method to learn Chinese idiom embeddings based on BERT. Our proposed method is adapted from a method for Chinese idiom recommendation (Tan and Jiang, 2020).

#### 4 Learning Chinese Idiom Embeddings

Continuous Bag-Of-Words (CBOW) and Skip-Gram with Negative-Sampling (SGNS) (Mikolov et al., 2013) are two most commonly used efficient log-linear prediction models for learning non-contextualized word embeddings. CBOW tries to predict a word based on its context, where the context is represented as the average word embeddings within the contextual window. In contrast, SGNS tries to predict the contextual words of a given word, and negative sampling is used to reduce the computational cost.

Both CBOW and SGNS have been used to learn Chinese word embeddings (Chen et al., 2015; Li et al., 2018). Chinese is an ideographic language with no explicit word delimiter between words (Li and Yuan, 1998). Chinese segmentation tools are
therefore used to identify word boundaries when learning Chinese word embeddings. On the other hand, Chinese words consist of characters, which have their own semantic meanings. Therefore character information has been incorporated to improve Chinese word embeddings (Chen et al., 2015). In addition, inspired by N-gram SGNS for English (Zhao et al., 2017; Bojanowski et al., 2017), which predicts contextual N-grams rather than contextual words, Li et al. (2018) trained Chinese word embeddings using N-gram SGNS and found that both N-gram and character features bring significant and consistent improvement.

However, Chinese idioms are not always treated as words by Chinese segmentation tools. They are sometimes separated into multiple words. Therefore, only a subset of the idioms in our idiom vocabulary can be found as words in existing pre-trained non-contextualized Chinese word embeddings, and we are only able to perform evaluation on this subset of idioms.

4.2 BERT and Its Variants

Recently, contextualized word embeddings have shown to be highly effective for many NLP tasks. BERT (Devlin et al., 2019) is probably the most commonly used contextualized word embedding model. The original BERT model is pre-trained using the Masked Language Model (MLM) task and the Next Sentence Prediction (NSP) task. Since the original BERT was proposed, there have been some variants of it proposed, including BERT with whole word masking (BERT-wwm) (Cui et al., 2019) and ERNIE (Zhang et al., 2019) that incorporates a multi-stage knowledge masking strategy which adds word-level masking, phrase-level masking and entity-level masking.

The original Chinese-BERT starts from embeddings of individual Chinese characters at the bottom layer. When BERT-wwm or ERNIE is applied to Chinese, although words are identified and masked using Chinese segmentation tools, the model still does not learn embedding vectors directly for entire words. Therefore, to obtain an embedding for an idiom, we need to aggregate the component characters’ embeddings. In this paper, we take the vector representations of individual characters at the top layer of BERT, and average these character representations as the embedding for the entire idiom.11

11We have also experimented with another setting where

![Figure 1: Model structure for BERT with SGNS. The red flow shows the path for the target idiom while the light blue flows show paths for negative sampled idioms used for the learning.](image)

4.3 Learning Idiom Embeddings with BERT

As we have pointed out earlier, existing non-contextualized Chinese word embedding methods model contextual words in a bag-of-word manner, which is suboptimal for encoding the contextual information. Chinese-BERT and its variants can better encode the contextual information using the Transformer architecture, but they do not learn a single embedding vector for an entire Chinese idiom, and therefore they are not ideal either because idioms often have non-compositional semantics. We propose to combine BERT contextual encoding with single embedding vectors for Chinese idioms.

Specifically, to train idiom embeddings, we perform the task of idiom prediction based on its context. Given an idiom \( v \) appearing in a context window \( c = (w_{-k}, \ldots, w_{-2}, w_{-1}, [\text{MASK}], w_1, w_2, \ldots, w_k) \), where \( w_i \) are the contextual words and [MASK] replaces the idiom \( v \) in the original text, the task aims to predict \( v \) based on \( c \). To do so, our idea is to assume that \( v \) has an embedding vector \( e_v \) to be learned. We then use BERT to derive a hidden representation \( h \) that represents \( c \) and use \( h \) and \( e_v \) to derive a log-linear score to indicate how likely \( v \) fits into the context \( c \).

Note that the task described above is similar to the prediction task used by CBOW, but instead of simply using the average word embedding to represent the context \( c \), our method uses BERT to encode \( c \). The task described above is also similar to the Masked Language Model task of BERT, but we mask and predict whole idioms rather than we use the [CLS] token’s representation at the top layer as the idiom representation. We found this to perform worse than using average character embedding.
individual characters.

Concretely, to use BERT to encode the sequence \(c\), following standard practice, we prepend the token [CLS] to the beginning of \(c\) and append [SEP] to the end of \(c\). We also include position embeddings. For segment embeddings, we treat the sequence \(c\) as a single segment. Let \(h_{\text{CLS}} \in \mathbb{R}^d\) denote the hidden vector produced by the last layer of BERT representing [CLS], and \(h_{\text{MASK}} \in \mathbb{R}^d\) the similarly produced hidden vector representing [MASK]. We then define the following vector \(h\) to combine \(h_{\text{CLS}}\) and \(h_{\text{MASK}}\) into a single vector representation because both are important for representing the context \(c\): 

\[
h = W h_{\text{CLS}}; h_{\text{CLS}} \odot h_{\text{MASK}}; h_{\text{CLS}} - h_{\text{MASK}},
\]

where \(\odot\) is element-wise multiplication between two vectors and \(W \in \mathbb{R}^{d \times 4d}\) is a matrix to be learned.

We then use a standard log-linear model based on the dot product between \(h\) and \(e_v\) to train our model. To use the hidden representation \(h\) of the context to predict the idiom \(v\), we take its idiom embedding \(e_v\), apply Layer Normalization (Ba et al., 2016) \(LN\) on it. We also adopt negative sampling to select negative Chengyu. The learning objective is defined as

\[
-(\log \sigma(LN(e_v)^	op h) + \sum_{v' \in N_v} \log (-LN(e_{v'})^	op h)),
\]

where \(N_v\) contains a fixed number of negative samples for each Chinese idiom, and \(\sigma(\cdot)\) is the sigmoid function. Besides the transformation \(W\) and \(LN\), during the training process, the BERT layers will be finetuned and the whole vocabulary will be learned from random initialization. The model structure is illustrated in Figure 1.

5 Experiments

5.1 Experiment Setup

Evaluation metrics: Recall that our main evaluation dataset is the ChIdSyn dataset that contains entries of query idioms and their near-synonyms, where these near-synonyms share at most one common character with the query idiom. We design two evaluation metrics to measure whether near-synonyms in ChIdSyn are close to each other in an embedding space. (1) \textit{Recall@K}: Given a query idiom \(v_n\), we rank all idioms based on their idiom embeddings’ cosine or Euclidean distances with the query idiom’s embedding. Let \(R^{(K)}_{v_n}\) represent the top-\(K\) ranked idioms. Let \(S_{v_n}\) denote the set of ground truth near-synonyms of \(v_n\). \textit{Recall@K} is defined as

\[
\text{Recall@K} = \frac{1}{N} \sum_{n=1}^{N} \frac{|S_{v_n} \cap R^{(K)}_{v_n}|}{|S_{v_n}|},
\]

where \(N\) is the total number of query idioms in ChIdSyn. (2) \textit{Coherence@K}: However, it is not guaranteed that all near-synonyms of a query idiom \(v\) are identified in the online resources we crawled, i.e., some of the top-\(K\) ranked idioms may be indeed near-synonyms but are not found in the ground truth near-synonym set. To overcome this limitation, we can measure whether a query idiom and its ground truth near-synonyms share many common “similar” idioms. In this way, even if a real near-synonym \(u\) of idiom \(v\) is missed from the ground truth, if \(u\) is found to be similar to both \(v\) and its ground truth near-synonyms, it will contribute positively to the metric. We therefore define the following metric, which we call \textit{Coherence@K}:

\[
\text{Coherence@K} = \frac{1}{N} \sum_{n=1}^{N} \frac{|\bigcap_{u \in S'_{v_n}} R^{(K)}_{u}|}{|\bigcup_{u \in S'_{v_n}} R^{(K)}_{u}|},
\]

where \(v_n\) is a query idiom, \(N\) is the total number of query idioms, \(S'_{v_n} = \{v_n\} \cup S_{v_n}\) (i.e., \(v_n\) together with its ground truth near-synonyms), and \(R^{(K)}_{u}\) is the top-\(K\) similar idioms to \(u\), where similarity can be based on either cosine or Euclidean distance.

Methods to be compared: We empirically compare the following embedding methods: (1) \textit{SGNS} and its variants: We use Chinese word embeddings released by Li et al. (2018), which are trained using the Skip-Gram with Negative Sampling method. There are a few variations of these embeddings. \textit{SGNS+B} uses bigram prediction, \textit{SGNS+C} incorporates character information, and \textit{SGNS+B+C} uses both bigram prediction and character information. Li et al. (2018) also experimented with different genres of text for training. In this paper, we use their pre-trained word embeddings trained on the literature genre because this provides fair comparison with our method, which is also trained on Chinese text in the literature genre. (2) \textit{BERT-wwm}: This refers to averaging the top-layer character representations after using the pre-trained Chinese-BERT-wwm (Cui et al., 2019) to process an idiom. (3) \textit{ERNIE}: This refers to averaging the top-layer character representations after using Chinese ERNIE (Zhang et al., 2019) to process an
We first present the results of all the methods we compare using the metrics Recall@K and Coherence@K on ChIdSyn, see Table 2. We can draw the following major conclusions from the table: (1) If we compare Ours-16 with the SGNS methods, we can see that Ours-16 clearly outperforms these SGNS methods. Recall that we use a similar context window size as the SGNS methods. The main difference of Ours-16 from the SGNS methods is that we use Chinese-BERT to encode the context whereas the SGNS methods do not model the interactions between the contextual words. This implies that when learning Chinese idiom embeddings, it is important to model the order of and interactions between the contextual words. (2) Comparing Ours-16 with BERT-wwm and ERNIE, we can see that Ours-16 also substantially outperforms these two BERT-based methods. Recall that the main difference of our method and these BERT methods is that we directly learn a single idiom embedding vector whereas for these BERT methods we need to aggregate character embeddings to derive idiom embeddings. The results suggest that many Chinese idioms’ semantic meanings cannot be simply derived from their character embeddings and therefore it is important to associate a Chinese idiom with a single embedding vector and to learn this embedding vector from the contexts of this idiom. (3) Ours-32 performs clearly better than Ours-16. This suggests that a larger context window is very useful for learning Chinese idiom embeddings, which have not been found to be the case for word embeddings (Lison and Kutuzov, 2017).

Besides the major conclusions drawn above, we can also see from the two tables that: (1) For the SGNS methods, adding character information may actually either hurt the performance or improve the performance very little. In other words, there is no consistent observation that character information helps for Chinese idiom embeddings, which is not the case for Chinese word embeddings (Chen et al., 2015; Zhao et al., 2017). This verifies our hypothesis that existing conclusions drawn from evaluating Chinese word embeddings may not apply to idiom embeddings. (2) For the two BERT-based methods, we can see that ERNIE performs clearly better than BERT-wwm. It is worth noticing that ERNIE uses Baidu Baike in which most idioms have entries and
would be treated as entities by the entity-level mask. Intuitively, the embeddings extracted using ERNIE should be better than BERT-WWM, whose CWS tools may not be able to recognize all the idioms.

5.3 Further Analysis

In this section, we conduct some further comparison and analysis using ChIdSyn-com and ChIdAnt.

Synonyms with Common Characters: Recall that we identified a set of near-synonyms that share two or more common characters. We suspect that these idiom synonyms are easier to be identified if the idiom embeddings rely more on character-level information. To verify this hypothesis, we compare the various methods using Recall@K based on cosine distance on ChIdSyn-com. The results are shown in Table 3. We can see that indeed those existing methods that rely more on character-level information, namely, SGNS+C, SGNS+B+C, BERT-wwm and ERNIE generally perform better than the other methods, including our methods. This verifies our hypothesis above. Note that because the synonyms in ChIdSyn-com share many common characters, being able to identify them does not imply that the embeddings truly capture the semantic meanings of the idioms. Since SGNS+C, SGNS+B+C, BERT-wwm and ERNIE actually do not perform well on ChIdSyn, we argue that they are effective only for synonyms sharing many common characters, and this implies that they rely on superficial patterns to encode idioms.

Antonyms: Recall that earlier we raised the hypothesis that good idiom embedding methods should be able to distinguish antonyms from synonyms, although both can be topically related to the query idioms. In fact, a previous study by Samenko et al. (2020) also found that embeddings contain information that distinguishes synonyms and antonyms. Inspired by them, we think that the separability of near-synonyms and antonyms may reflect the quality of the learned embeddings. We therefore visualize the distributions of cosine distances (i.e., 1 minus cosine similarity) of idiom near-synonym pairs and antonym pairs in Figure 2, using ChIdSyn and ChIdAnt. We can see from the figure that our methods Ours-16 and Ours-32 clearly has a distinguishable cosine distance distribution for antonyms compared with synonyms, whereas for the other methods the two distributions are less distinguishable. This again demonstrates the advantage of our idiom embedding methods.

6 Conclusion

In this paper, we constructed a new evaluation dataset that contains Chinese idiom synonyms and antonyms to facilitate the evaluation of Chinese idiom embeddings. We presented a method that learns Chinese idiom embeddings by predicting idioms based on BERT-encoded contexts. We also propose two metrics to measure closeness of synonyms in the embedding space. Our method performs substantially better than existing methods.

<table>
<thead>
<tr>
<th>$K$</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGNS</td>
<td>0.130</td>
<td>0.223</td>
<td>0.270</td>
<td>0.334</td>
</tr>
<tr>
<td>SGNS+B</td>
<td>0.175</td>
<td>0.287</td>
<td>0.341</td>
<td>0.404</td>
</tr>
<tr>
<td>SGNS+C</td>
<td>0.518</td>
<td>0.775</td>
<td>0.857</td>
<td>0.924</td>
</tr>
<tr>
<td>SGNS+B+C</td>
<td>0.526</td>
<td>0.776</td>
<td>0.846</td>
<td>0.908</td>
</tr>
<tr>
<td>BERT-wwm</td>
<td>0.467</td>
<td>0.662</td>
<td>0.714</td>
<td>0.786</td>
</tr>
<tr>
<td>ERNIE</td>
<td>0.531</td>
<td>0.760</td>
<td>0.825</td>
<td>0.880</td>
</tr>
<tr>
<td>Ours-16</td>
<td>0.380</td>
<td>0.555</td>
<td>0.612</td>
<td>0.675</td>
</tr>
<tr>
<td>Ours-32</td>
<td>0.449</td>
<td>0.655</td>
<td>0.722</td>
<td>0.786</td>
</tr>
</tbody>
</table>

Table 3: Recall@K on ChIdSyn-com.

Figure 2: Cosine distance distribution of near-synonym and antonym pairs.
References


Siyu Long, Ran Wang, Kun Tao, Jiali Zeng, and Xinyu Dai. 2020. Synonym knowledge enhanced...


Abstract

Understanding idioms is important in NLP. In this paper, we study to what extent a pre-trained BERT model is able to encode the meaning of a potentially idiomatic expression (PIE) in a certain context. We make use of a few existing datasets and perform two probing tasks: PIE usage classification and idiom paraphrase identification. Our experiment results suggest that BERT indeed is able to separate the literal and idiomatic usages of a PIE with high accuracy. It is also able to encode the idiomatic meaning of a PIE to some extent.

1 Introduction

Understanding idiomatic expressions is important for NLP tasks such as sentiment analysis (Balahur et al., 2010; Williams et al., 2015) and machine translation (Isabelle et al., 2017; Shao et al., 2018). However, due to the non-compositionality of idioms, it remains a challenge to model the semantic meanings of idioms effectively (Sag et al., 2002; Shwartz and Dagan, 2019).

BERT is a contextualized pre-trained language model that has been widely used and proven to be highly effective for many NLP tasks (Devlin et al., 2019). To better understand how BERT works, recently the community has adopted the approach of probing, where a probing task is designed to test whether BERT encodings contain sufficient information to perform the task well. Examples of probing tasks include POS tagging and parsing (Hewitt and Liang, 2019; Wu et al., 2020) as well as semantic reasoning tasks such as understanding numbers (Wallace et al., 2019).

It is therefore also natural to ask whether BERT encodes any knowledge about the usage and meanings of idioms, given that BERT was trained on huge corpora, which must contain many idiomatic expressions. However, this problem has not been well explored. To the best of our knowledge, the closest existing work is by Shwartz and Dagan (2019), who studied whether pre-trained (static and contextualized) word embeddings can detect meaning shift and implicit information of phrases, with the help of several probing tasks. However, we believe there is a need for further exploration. We note that Shwartz and Dagan (2019) did not specifically focus on idioms; only one of the six probing tasks was directly related to idioms, and only idiomatic noun compounds were studied. Since English idioms have different syntactic structures, it would be useful to experiment with a higher coverage of different types of idioms.

In this paper, we focus on probing BERT to understand whether BERT embeddings can encode the meanings of a diverse range of different types of idioms. We propose two probing tasks to test whether BERT understands idioms. First, given a context containing a potentially idiomatic expression (PIE), the task is to decide whether the meaning of the PIE is literal or idiomatic, based on the BERT-encoded contextualized embedding of the PIE. We hypothesize that if pre-trained BERT could perform the task well, it would indicate that BERT knows the difference between literal and idiomatic usages of the same expression based on its context. For this task, we use a large dataset recently released by Haagsma et al. (2020), which covers 1756 unique idioms and 50K contextual sentences, much larger and more diverse than the idiomatic noun compounds dataset used by Shwartz and Dagan (2019). However, this task is not sufficient to show whether BERT truly understands the idiomatic meaning of a PIE. In order to test this, we design a second probing task based on existing idiom paraphrase datasets. The task is to select the correct paraphrase of an idiom among a set of candidate phrases based on the cosine similarity between the idiom’s BERT embedding and these...
candidate phrases’ BERT embeddings. We hypothesize that if the correct paraphrase could be ranked higher than other irrelevant phrases, it would indicate that BERT indeed understands the idiomatic meaning of the idiom.

It is important to note that our objective is not to improve the performance of the two tasks by designing effective learning methods; rather, the objective is to use these two tasks to probe pre-trained BERT in order to understand how much BERT encodes the meanings of idioms. Therefore, the models for the two probing tasks are simple models without many parameters to be learned.

Through our empirical study using both the original BERT and ERNIE2 (Sun et al., 2020) (an improved version of BERT), we find that compared with non-contextualized embedding representations of PIEs, contextualized BERT and ERNIE2 embeddings of PIEs can clearly achieve higher accuracy for PIE usage classification, with an accuracy level around 90%, suggesting that BERT can use the context to accurately guess whether an expression is used literally or idiomatically. For paraphrase identification, we find that BERT and ERNIE2 perform significantly better than a random baseline, although the absolute performance is still considered low. Since paraphrase identification is itself challenging, to put things in perspective, we also compare with paraphrase identification for general multi-word expressions (MWEs). Contrary to our expectation, we find that identifying paraphrases for general MWEs does not necessarily fare better than for idioms. Further analysis reveals that this is because BERT contextualization actually hurts paraphrase identification for general MWEs but not so for idioms.

2 Related Work

2.1 Probing Tasks

The notion of probing (Ettinger et al., 2016) or a probing task (Conneau et al., 2018) refers to the use of a classification problem to reveal whether certain linguistic properties of sentences are captured in the input embedding representations of the sentences fed into the classification model. There have been studies investigating what properties of a sentence its embedding might have contained (Ettinger et al., 2016; Shi et al., 2016; Adi et al., 2017). The properties being probed include semantic roles (Ettinger et al., 2016), negation scopes (Ettinger et al., 2016), constituents (Shi et al., 2016), part-of-speech tags (Shi et al., 2016), sentence lengths (Adi et al., 2017), word orders (Adi et al., 2017), agreement information (Giulianelli et al., 2018) and tense of the main clause (Bacon and Regier, 2018). With the emergence of contextualized embeddings such as BERT (Devlin et al., 2019) and ELMO (Peters et al., 2018a), researchers have also applied probing tasks to word-level contextual representations (Tenney et al., 2019), attention mechanisms (Clark et al., 2019) and syntactic knowledge (Peters et al., 2018b; Hewitt and Manning, 2019). Probing phrasal representations to study lexical composition has also attracted attention. Jawahar et al. (2019) found that the compositional scheme underlying BERT mimics classical, tree-like structures. Shwartz and Dagan (2019) conducted a series of experiments and concluded that lexical composition can shift the meanings of the constituent words and introduce implicit information. Yu and Ettinger (2020) reminded us that phrase representation in transformer models still relies heavily on word content, with little evidence of sophisticated composition of phrase meaning like that done by humans. Our work differs from these existing studies in that we focus on idiomatic expressions rather than any phrases, and we use a recently released large-scale idiom dataset to facilitate our study.

2.2 Potentially Idiomatic Expressions

Potentially Idiomatic Expressions (PIEs) originate from multiword expressions (MWEs) which have both an idiomatic interpretation and a literal interpretation, for example, *spill the beans*. Identifying the correct meaning of a PIE in a certain context is crucial for many downstream tasks including sentiment analysis (Williams et al., 2015), automatic spelling correction (Horbach et al., 2016) and machine translation (Isabelle et al., 2017). There has been both supervised (Sporleder and Li, 2009) and unsupervised (Haagsma et al., 2018; Kurfalı and Östling, 2020) approaches to solve this problem. For example, Feldman and Peng (2013) treated idiom recognition as outlier detection, which does not rely on costly annotated training data. Peng et al. (2014) incorporated the affective hypothesis of idioms to facilitate the identification of idiomatic operations. Different from these studies, our object is not to improve the performance of idiom recognition but rather to use the task as a probing task to understand the capabilities of BERT to en-
code idioms. With newly created large scale dataset MAGPIE (Haagsma et al., 2020), we can further investigate how contextualized word representations works for idiomatic expressions and literal ones.

2.3 Paraphrase Identification

Paraphrase identification aims to determine whether a pair of language units such as sentences have the same meaning (Kauchak and Barzilay, 2006) or whether a given paraphrase candidate can replace a given language unit in its context without changing overall semantic meaning of the text (Yimam et al., 2016). Idiom paraphrasing is a challenging task that has been attracting continuous attention from the community. For example, Liu and Hwa (2016) investigated the effectiveness of a phrasal substitution method to replace idioms with literal expressions, indicating that high quality paraphrasing of idiomatic expressions can be achieved. Yimam et al. (2016) researched a paraphrase-scoring annotation task and showed that the contexts have an impact on the ranking of paraphrases. Haagsma et al. (2018) looks at the literal representation of the PIE’s figurative sense (similar to dictionary definitions of an idiom’s meaning, which can also be treated as paraphrase) to facilitate potentially idiomatic expression classification. Different from the studies above, in this paper, we focus on understanding whether pre-trained BERT models encode the semantic meanings of idioms, using idiom paraphrase identification as the probing task. We adopt three phrase-level paraphrase datasets for our probing task. Using this task, we probe how contextualization in transformers may affect the semantic relatedness of phrases.

3 Probing Tasks

We design two probing tasks to answer two research questions: (1) Can BERT distinguish the idiomatic usage of a PIE from its literal usage? (2) Can BERT understand the idiomatic meaning of an idiom? Both questions are related to the capabilities of BERT to understand idioms, but the second task is more demanding than the first. The two tasks also share similar objectives as the probing tasks designed by Shwartz and Dagan (2019), which aimed to test whether pre-trained word embeddings can detect the shift of meaning of a phrase from its component words, and whether pre-trained word embeddings understand the implicit meaning of a phrase. However, they are conducting probing at word level, which focuses on whether the meaning of a word in a noun compound (NC) is literal. The dataset only has 90 noun compounds (Reddy et al., 2011). Although they try to augment the dataset using Tratz (2011), the dataset is still limited to 3K. The paraphrase identification task used by them also uses compounds and addresses whether the paraphrase describes the semantic relation between two words of a noun compound (Hendrickx et al., 2013).

In this paper, we use a much larger dataset called MAGPIE (Haagsma et al., 2020) that covers much more potentially idiomatic expressions for phrase-level literal-idiomatic classification. To make the task more challenging, we choose to split the data such that the idiomatic expressions in the training, development, and test sets do not overlap. We further adapt several paraphrase datasets (Liu and Hwa, 2016; Yimam et al., 2016; Pershina et al., 2015) to compare phrasal semantic relatedness for idioms. We compare the effect of BERT encodings at different layers for the two probing tasks to better understand the effect of contextualization.

3.1 PIE Usage Classification

Many MWEs can be interpreted either literally or idiomatically. In some literature, these expressions are defined as potentially idiomatic expressions (PIEs) (Sporleder and Li, 2009; Haagsma et al., 2018, 2020). For example, “spill the beans” can either be used literally to refer to the action of spilling beans or in its idiomatic sense to refer to disclosing some secrets. However, current approaches are investigating this problem with the limitation to one or more syntactic patterns. In this paper, we propose to use the latest large scale dataset MAGPIE to probe how BERT is capturing the difference of literal and non-literal usage of a PIE.

Task Definition. Given a piece of context denoted as \( w_1, w_2, \ldots, w_n \) containing a PIE with \( m \) words, \( w_i, \ldots, w_{i+m-1} \), the task is to decide whether the PIE is used with its literal meaning or its idiomatic meaning. Performance is measured by accuracy. It is important to note that since our goal is to test whether pre-trained BERT can already encode such knowledge, we do not train a classifier per idiom. Instead, we train a single binary classifier using a set of training PIEs and their labeled contexts, and test the classifier on a separate set of different test PIEs and their contexts.
Data. We use the MAGPIE dataset (Haagsma et al., 2020), which is the largest-to-date corpus of English PIEs and labeled instances of both their literal and idiomatic usages in different contexts. The corpus comprises 1756 unique PIEs and more than 50K contexts, an order of a magnitude larger than previous similar resources. Annotations of MAGPIE included various aspects: annotation (dis)agreement, distribution of idiom types, sense distributions across types, composition of the ‘other’-category, and influence of genre. An example of MAGPIE is given in Table 1. In this paper, we further analyse what might be the reason of BERT’s advantage in connection with annotation agreement.

Table 1: An example from MAGPIE dataset with details of annotations.

| Context: | Think of a sunflower turning its flower head towards a source of light — and therefore of energy. The sunflower does not learn by experience to turn its head more effectively as it matures, or not to turn at all if it is repeatedly electrically shocked every time it does so. |
| Annotation: | Label: literal | PIE: turn head |
| | Confidence: 0.75 | Genres: W nonAc: nat science |
| | Judgment Count: 4 | |
| | Variant Type: combined-inflection | Label Distribution: {‘idiomatic’: 0.25, ‘literal’: 0.75} |

3.2 Idiom Paraphrase Identification

In this paper, to further understand whether BERT has learned the idiomatic meaning of phrases, we propose the Idiom Paraphrase Identification probing task to check whether contextualized representations of PIEs encoded by BERT have shifted meanings that are closer to their paraphrases.

Task Definition. Given a piece of context denoted as \((w_1, w_2, \ldots, w_n)\) containing a PIE \(w_i, \ldots, w_{i+m-1}\) where the PIE is known to be used idiomatically, and given a set of candidate phrases \(\mathcal{P} = \{p_1, p_2, \ldots, p_k\}\), where each \(p_i \in \mathcal{P}\) is a MWE and one of them is a paraphrase of the given idiom, the task is to identify the correct paraphrase from \(\mathcal{P}\). We cast this task as a ranking problem and use Mean Reciprocal Rank (MRR) to measure the performance.

Data. We combine different resources described below to create the data needed to perform this paraphrase identification task. Specifically, we create three datasets: (1) Idioms-MWEs, (2) MWEs-MWEs, and (3) Idioms-Idioms. Details of the collection of these three datasets are listed below:

- **Idioms-MWEs**: We use the idiom paraphrase dataset created by Liu and Hwa (2016). Each instance in this dataset is a context sentence containing an idiom together with a phrase that can substitute the idiom in the context. The dataset was created by shortening the definitions of these idioms from a dictionary and performing appropriate grammatical and referential transformations to ensure that the idiom substitution fits seamlessly into the original context. The paraphrases have also been verified and refined by human annotators. This gives us a dataset with high quality paraphrases of idiomatic expressions. The dataset contains 171 unique idioms, each with a single context sentence and a paraphrase.

- **MWEs-MWEs**: Since paraphrase identification itself is likely a challenging task even for non-idiomatic MWEs, in order to put things in perspective, we also make use of another paraphrase dataset that contains pairs of MWEs that are paraphrases. Yimam et al. (2016) investigated the impact of context for the paraphrase ranking task using both multi-word expressions and single words. The dataset covers 17k data points (2k MWEs and 15k single word) annotated through crowd-sourcing. The 2k MWEs are of particular interest to us in this probing task. We processed the original dataset by retaining only those paraphrase pairs with a human agreement score of 4, which gives us a final set of 176 entries of a MWE in a context as well as their paraphrases. We find that these 176 entries do not overlap with the PIEs in the MAGPIE dataset, suggesting that these MWEs are likely all non-idiomatic expressions. By performing paraphrase identification on this dataset, we can get a sense of the expected performance for paraphrase identification on phrases that are not idiomatic.

- **Idioms-Idioms**: Pershina et al. (2015) presented idiomatic expressions as a new domain for short-text paraphrase identification and
This Cuban Black Bean recipe is pretty much as easy as beans get and they are SO delicious.

If only I could soup up this computer to run just a little faster.

She constantly complains of boredom as her presence at home is merely decorative, while her husband is heavily involved in his scholarly interests.

Table 2: Paraphrase evaluation datasets. We select one example from each dataset.

<table>
<thead>
<tr>
<th>Size</th>
<th>Sentence</th>
<th>Paraphrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idioms-Idioms 158</td>
<td>This Cuban Black Bean recipe is pretty much as easy as beans get and they are SO delicious.</td>
<td>piece of cake</td>
</tr>
<tr>
<td>Idioms-MWEs 171</td>
<td>If only I could soup up this computer to run just a little faster.</td>
<td>increase the power of</td>
</tr>
<tr>
<td>MWEs-MWEs 176</td>
<td>She constantly complains of boredom as her presence at home is merely decorative, while her husband is heavily involved in his scholarly interests.</td>
<td>in her house</td>
</tr>
</tbody>
</table>

For each dataset, we list its size and one example in Table 2.

To create the set of candidate paraphrases, we simply pool the paraphrases of all the entries of the three datasets together as the set of candidate paraphrases for all instances.

4 Experiments

For each of the two probing tasks above, we use pre-trained BERT\(^2\) and ERNIE2\(^3\) to process each context \((w_1, w_2, \ldots, w_n)\). Following standard practice, we prepend the [CLS] token to the beginning of the sequence and append the [SEP] token to the end. The sequence is then fed into an L-layer BERT. Let \(h^k_i \in \mathbb{R}^d\) denote the hidden vector produced by the \(k\)th layer of BERT representing \(w_i\). When \(k = 0\), \(h^0_i\) denotes the combined representation of the word embedding, the position embedding and the token type embedding before it is fed into the transformer-based encoder.

For each PIE, we get a sequence of hidden vectors at the \(k\)th layer for the \(m\) tokens inside this PIE as follows: \(p^k = (h^k_i, h^k_{i+1}, \ldots, h^k_{i+m-1})\). We will use these contextualized BERT embeddings of the PIE as input to the model for the probing tasks. Note that when training the model for a probing task, BERT is not fine-tuned.

For both probing tasks, we experiment with both the original BERT (Devlin et al., 2019) and ERNIE2 (Sun et al., 2020), which supports phrase masking by using lexical analysis and chunking tools to get the boundary of phrases in the sentences. Our code and data are released on github\(^4\).

4.1 PIE Classification

After we get the hidden representation \(p^k = (h^k_i, h^k_{i+1}, \ldots, h^k_{i+m-1})\) of the PIE, we further encode the sequence into a single vector using a bidirectional LSTM encoder. We then treat this vector as input to train the binary PIE usage classifier using a linear classifier.

We show the accuracy of the trained PIE usage classifier on both the development set and the test set in Table 3. We include a baseline BL-majority that always predicts the usage to be idiomatic. This is because we observe that there are more instances in this dataset labeled as idiomatic than literal. We also include another baseline BL-GloVe, which uses the static GloVe word embeddings (Pennington et al., 2014) to replace the BERT encoded representations. For BERT embeddings, we include

\(^2\)huggingface.co/bert-base-uncased

\(^3\)huggingface.co/nghuyong/ernie-2.0-en

\(^4\)https://github.com/VisualJoyce/CiYi
the results using the bottom layer (Layer-0) and the results using the final layer (Layer-12). Including Layer-0 is for us to observe how the static embeddings of BERT have performed.

From the table, we can draw the following conclusions: (1) The baseline method **BL-majority** achieves an accuracy above 50%. This shows that the dataset is not balanced, with more instances of idiomatic usage. (2) Using Layer-0 of BERT and ERNIE2, i.e., using only static word embeddings, we can see that the performance is always above 80% and is very close to **BL-GloVe**. This suggests that even the static word embeddings contain some prior knowledge about whether the expression is literal or idiomatic. (3) Using Layer-12 of BERT and ERNIE2, we can see that the accuracy of PIE usage classification significantly increased compared with using Layer-0. In fact the absolute accuracy level is quite high, reaching 90%. This confirms that with BERT contextualization, the embeddings of the PIE better reflect the usage of the PIE, allowing the classifier to easily predict whether the PIE is used literally or idiomatically. This shows that BERT can indeed encode the knowledge about the usage of a PIE.

<table>
<thead>
<tr>
<th></th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL-majority</td>
<td>71.76</td>
<td>68.78</td>
</tr>
<tr>
<td>BL-GloVe</td>
<td>80.52</td>
<td>82.05</td>
</tr>
<tr>
<td>BERT Layer-0</td>
<td>83.90</td>
<td>81.28</td>
</tr>
<tr>
<td>BERT Layer-12</td>
<td>90.33</td>
<td>91.67</td>
</tr>
<tr>
<td>ERNIE2 Layer-0</td>
<td>84.65</td>
<td>81.98</td>
</tr>
<tr>
<td>ERNIE2 Layer-12</td>
<td>89.03</td>
<td>92.11</td>
</tr>
</tbody>
</table>

Table 3: PIE classification accuracy.

Given the large gap between the classification accuracy using Layer-0 and Layer-12, next we exper-
iment with other intermediate layers of the Transformer architecture for BERT and ERNIE2. The results are shown in Figure 1a. From the figure we find that starting from around Layer-4 the performance stabilizes and the last layer is not necessarily the one with the best performance. This shows that BERT requires just a few rounds of contextualization to encode the idiom usage information.

To better understand how BERT contextualization improves PIE usage classification, we further zoom into the two different types of errors: (1) literal usage mistakenly classified as idiomatic usage, and (2) idiomatic usage mistakenly classified as literal usage. We show the numbers of these error cases in four confusion matrices in Figure 1b (one confusion matrix for one of Layer-0, Layer-4, Layer-8 and Layer-12), where the lower-left corner shows the first type of errors and the upper-right corner shows the second type of errors. In Figure 2, we further show the precision, recall and F1 scores across all the layers by either choosing idiomatic or literal as the positive label. We observe that interestingly the error reductions from Layer-0 to Layer-12 comes mostly from the group literal-idiomatic where literal expressions are wrongly predicted to be idiomatic. We hypothesize that this is because without contextualization, some of the words in these PIEs tend to indicate that the PIEs are used idiomatically, probably because these words have appeared often in other idiomatic expressions in the training data; but after considering the specific contexts these PIEs are placed in, i.e., with BERT contextualization, the model recognizes that these contexts are semantically similar to the literal meanings of the tokens inside these PIEs, and therefore predict the usage as being literal. This shows that with more contextualization, BERT embeddings help the most in recognizing literal usages of PIEs.

We further ask the question whether those instances where BERT embeddings did not do well for the PIE usage classification task are those instances where human annotators’ agreement is also low. To answer this question, we show the average annotation agreement scores on the test set for correctly predicted instances and incorrectly predicted instances. The statistics are shown in Figure 3. The red line shows the average agreement score over all test instances, the green line shows the average agreement score over those instances whose ground truth labels are “idiomatic”, and the blue line shows the average agreement score over those instances with the ground truth label “literal”. We can see that human annotations have a clearly higher degree of agreement on those idiomatic usages of PIEs, but a lower agreement when a PIE is likely used literally. The four bars in Figure 3 shows the average agreement scores of correctly and incorrectly predicted instances, grouped by the ground truth labels. We can see that clearly those incorrectly predicted instances (shown in light gray bars) have clearly lower human agreement scores compared with the correctly predicted ones. This verifies our hypothesis that the model tends to make mistakes on those instances which humans also find hard.

### 4.2 Paraphrase Identification

<table>
<thead>
<tr>
<th></th>
<th>Idioms</th>
<th>MWEs</th>
<th>Idioms</th>
<th>MWEs</th>
<th>MWEs</th>
<th>Idioms</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL-random</td>
<td>0.013</td>
<td>0.013</td>
<td>0.013</td>
<td>0.013</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BERT</td>
<td>0.163</td>
<td>0.104</td>
<td>0.154</td>
<td>0.154</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERNIE2</td>
<td>0.202</td>
<td>0.078</td>
<td>0.136</td>
<td>0.136</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: MRR scores for paraphrase ranking.

For the paraphrase identification task, after we get the hidden representation $p_k$ of the PIE in its context, we take the average of these vectors to obtain a single vector. For each candidate paraphrase, we perform the same encoding, without any context, and then take the average of the produced hidden vectors. Finally, we rank the candidates based on the cosine similarity between the PIE’s embedding and the candidate’s embedding.

The Mean Reciprocal Rank (MRR) scores are
listed in Table 4. For comparison, we consider a baseline that randomly ranks the candidates. We can observe the following from the table: (1) BERT and ERNIE2 can perform better than the random baseline on Idioms-MWEs, although the absolute values of MRR are low. This shows that BERT contextualized embeddings can still encode the idiomatic meanings of idioms to some extent. (2) We also observe that identifying paraphrases for general multi-word expressions (MWEs-MWEs), which are likely not idiomatic, is not easier than for idioms. This is counter-intuitive and we will show further investigation below. (3) Identifying paraphrase idioms of idioms (Idioms-Idioms) is a bit harder than identifying general multi-word-expression-based paraphrases. This maybe because the candidate idioms are not contextualized, and therefore their embeddings do not reflect their idiomatic meanings.

To better understand why paraphrase identification for general MWEs has even lower performance than for idioms, we again test the performance using different layers of BERT/ERNIE2 embeddings. The results are shown in Figure 4a. Now it is clear that with non-contextualized embeddings (i.e., Layer-0), paraphrase identification for general MWEs is actually much easier than for idioms. This is intuitive because the meaning of non-idiomatic MWEs can be derived from their component words and therefore contextualization is not needed. The figure also shows that with more contextualization, performance of paraphrase identification for general MWEs is largely hurt, but this is not the case for idioms. It’s also interesting that, for Idioms-Idioms, the MRR scores do not change much with layers. We think this may due to both an idiom and its idiomatic paraphrase share less overlap with the context.

Noticing that the performance of paraphrase identification for Idioms-MWEs surpasses MWEs-MWEs at Layer-8, i.e., when there is some degree of contextualization, we conduct some further analysis to understand why. Specifically, given a query idiom (or query MWE) $q$, its context $c$, and its ground truth paraphrase MWE $p$, we would like to check if $p$ tends to have common words with $q$ and $c$, respectively. Our hypothesis is that if $p$ shares common words with $c$, then contextualized word embeddings are helpful because they encode the context $c$. We show our analysis in Figure 4b. In the left hand side of the figure, the light gray bar shows the percentage of test instances in the MWEs-MWEs dataset where the query MWE $q$ shares at least one common word with the ground truth paraphrase $p$, and the dark gray bar shows the percentage of test instances in MWEs-MWEs where the context $c$ shares at least one common word with the ground truth paraphrase $p$. The right hand side of the figure shows the same percentages for the Idioms-MWEs dataset. We can see that for MWE-MWE paraphrase pairs, it is less...
common for the ground truth paraphrase to share a common word with the context of the query phrase, compared with Idiom-MWE paraphrase pairs. This is reasonable because for an idiom, its idiomatic meaning is often not directly linked to the semantic meanings of their component words, and therefore words in the idiom itself may not overlap with words in its paraphrase; on the other hand, the context where an idiom appears may imply the idiom’s idiomatic meaning, and therefore may have word overlap with the paraphrase. The statistics shown in Figure 4b shows that because for MWEs, their paraphrases are less likely to share common words with the contexts where the MWEs appear, contextualization done by BERT therefore not only is not so useful but also may harm the performance of paraphrase identification.

5 Conclusion

In this paper, we use two probing tasks to study whether BERT understands English idioms. In conclusion, we find that BERT is able to detect idiomatic usages of a PIE with a high accuracy, and with more contextualization as the layer increases, BERT helps the most in recognizing literal usages of PIEs. However, this only proves that BERT is effective in detecting meaning shift for idiomatic expressions. To further probe if the shifted meanings are closer to their paraphrases, we adopt the paraphrase identification task by gathering three different types of paraphrase pairs, MWEs-MWEs, MWEs-Idioms and Idioms-Idioms. Our experiments show that BERT is able to encode the idiomatic meaning to some extent. However, contextualization may have different effects for MWEs and idioms, which still requires further exploration to fully explain.

References


Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does bert look at? an analysis of bert’s attention. In Black-BoxNLP@ACL.


Hessel Haagsma, Malvina Nissim, and Johan Bos. 2018. The other side of the coin: Unsupervised disambiguation of potentially idiomatic expressions by


Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword


An Empirical Analysis of Topic Models:
Uncovering the Relationships between Hyperparameters,
Document Length and Performance Measures

Silvia Terragni, Elisabetta Fersini
University of Milano-Bicocca
Viale Sarca 336, 20126
Milan, Italy
s.terragni4@campus.unimib.it, elisabetta.fersini@unimib.it

Abstract

Neural Topic Models are recent neural models that aim at extracting the main themes from a collection of documents. The comparison of these models is usually limited because the hyperparameters are held fixed. In this paper, we present an empirical analysis and comparison of Neural Topic Models by finding the optimal hyperparameters of each model for four different performance measures adopting a single-objective Bayesian optimization. This allows us to determine the robustness of a topic model for several evaluation metrics. We also empirically show the effect of the length of the documents on different optimized metrics and discover which evaluation metrics are in conflict or agreement with each other.

1 Introduction

Topic models (Blei, 2012) are probabilistic generative models that aim at identifying the underlying themes, or topics, in a collection of documents. Although they are used in a vast range of applications, from text exploratory purposes to information retrieval tasks (Boyd-Graber et al., 2017), most of the investigations disregard the elements that influence the results generated by the models and, in particular, what is the effect on their performance.

Several works explore topic modeling over a range of different models, topics, and measures, but usually focus on classical topic models (Greene et al., 2014; Stevens et al., 2012), e.g. Latent Dirichlet Allocation (LDA) (Blei et al., 2003) or Non-negative Matrix Factorization (NMF) (Lee and Seung, 2000), and solely on a single evaluation measure (Stevens et al., 2012; O’Callaghan et al., 2015).

Doan and Hoang recently made an effort to benchmark neural topic models, however, the authors seem to disregard the importance of the hyperparameter selection. In fact, the evaluations of topic models are usually limited to the comparison of models whose hyperparameters are fixed. Yet, the hyperparameters that control the models can have a great impact on their performance. Therefore, fixing them prevents the researchers from discovering the best topic model on a given dataset. In the latest years, Neural Topic Models (NTM) (Zhao et al., 2021; Dieng et al., 2020; Bianchi et al., 2021a,b) have gained popularity, due to their flexibility and scalability. The problem of finding the best hyperparameter configuration has become even more compelling, since topic models based on neural networks are usually controlled by a high number of hyperparameters.

In this paper, we perform an empirical analysis of recent NTMs by optimizing the hyperparameters of the models with respect to different metrics. We aim to investigate if there exists a potential relationship between hyperparameters, document length and performance measures, to finally understand under which conditions we can exploit at best the potentiality of each model. In particular, the following research questions have been addressed:

RQ1: To what extent are Neural Topic Models robust across different evaluation metrics?
RQ2: Does the document length affect the Neural Topic Models on different performance measures?
RQ3: Does the optimization of a model’s hyperparameters for a given performance metric imply good performance on other measures?

To this purpose, we use Bayesian Optimization (BO) (Archetti and Candelieri, 2019), a well-known and efficient strategy for hyperparameter tuning, to determine the optimal hyperparameter settings for four different evaluation metrics of five state-of-the-art NTMs. The hyperparameter optimization allows us to guarantee a fair comparison.
between the models and investigate their behavior with different hyperparameter settings.

2 Methodology

In this work, we conduct an empirical comparison of different state-of-the-art NTMs. We adopt a single-objective Bayesian Optimization approach, using the comparative framework topic modeling OCTIS (Terragni et al., 2021a), to optimize the hyperparameters of five different topic models with respect to four different evaluation metrics. Each metric investigates a different aspect of a model. Bayesian Optimization (Archetti and Candelieri, 2019; Kandasamy et al., 2020) is a Sequential Model-Based Optimization (SMBO) strategy that allows us to optimize all the hyperparameters by treating the topic model as a black-box function. The model is in fact viewed just in terms of its input (the hyperparameters) and its output (the distribution of the topics over the vocabulary and the topic distribution for each document).

BO uses the model’s configurations evaluated so far to approximate the value of the performance metrics with respect to the model’s hyperparameters and then selects a new promising configuration to evaluate. The two key components are the probabilistic surrogate model aimed at approximating the performance metrics to optimize, and the acquisition function that uses the mean of the surrogate model and the confidence (i.e. its standard deviation) to select the next configuration.

Optimizing a model’s hyperparameters not only allows us to investigate the robustness of a model over different evaluation metrics (RQ1), but we can also investigate the performance of the optimized evaluation metric on datasets with different features (RQ2) and the relationship between the optimized evaluation metric and the other metrics (RQ3).

2.1 Topic Models

In our investigation, we focus on the following recent state-of-the-art topic models based on a neural variational frameworks. We consider Neural LDA (Srivastava and Sutton, 2017, NeurLDA), Product-of-experts LDA (Srivastava and Sutton, 2017, ProdLDA), the Embedded Topic Models (Dieng et al., 2020, ETM), and finally we use a variant of the family of Contextualized Topic Models, namely the Zero-shot Contextualized Topic Model (Bianchi et al., 2021b, CTM).

All these neural models are based on the Variational Autoencoder (VAE) presented in Miao et al. The neural variational framework trains an inference network to map the bag-of-words (BoW) document representation into a continuous latent representation. A decoder network reconstructs the BoW by generating its words from the document representation. This document representation is $K$-dimensional, where $K$ is the number of topics.

NeurLDA and ProdLDA (Srivastava and Sutton, 2017) explicitly approximate the Dirichlet prior using Gaussian distributions, instead of using a Gaussian prior. In addition, ProdLDA replaces the word-topic distribution in LDA with a product of experts (Hinton, 2002).

CTM (Bianchi et al., 2021b) extends ProdLDA by replacing the BoW document representation of the input with the corresponding pre-trained contextualized representations of the documents. These representations derive from contextualized language models, e.g. BERT (Devlin et al., 2019).

Concerning ETM (Dieng et al., 2020), words and topics are represented in the same embedding space. The word-topic distribution is proportional to the exponentiated inner product of the topic embedding and each word embedding. ETM can automatically learn the word embedding representations or use pre-trained word embeddings. Following the original paper, we will refer to the former version of the model as ETM, while the one that uses pre-trained word embeddings (PWE) will be referred to as ETM-PWE.

We also consider the well-known Latent Dirichlet Allocation (Blei et al., 2003, LDA) as a baseline. LDA is a probabilistic model that describes a corpus through $K$ topics, seen as distributions of words over a vocabulary $W$. A document is assumed composed of a mixture of topics following a Dirichlet distribution, where a topic drawn from the mixture is assigned to each word of the documents.

2.2 Evaluation Metrics

We consider four evaluation metrics that investigate different aspects of a topic model.

F1 refers to the Micro-F1 measure, the weighted average of the F1 measure for each class. We train a linear support vector machine (SVM) that predicts the document’s class using the topic distribution $\theta$ of each document (given by each topic model) as
its feature representation (Terragni et al., 2020a).

**IRBO** (Bianchi et al., 2021a; Terragni et al., 2021b) is a measure of topic diversity (0 for identical topics and 1 for completely different topics). It is based on the Ranked-Biased Overlap measure (Webber et al., 2010). Topics with common words at different rankings are penalized less than topics sharing the same words at the highest ranks.

**NPMI** (Lau et al., 2014) measures Normalized Pointwise Mutual Information of each pair of words \((w_i, w_j)\) in the 10-top words of each topic. It is a topic coherence measure, that evaluates how much the words in a topic are related to each other.

**KL-B** (AlSumait et al., 2009; Terragni et al., 2020b) is the Kullback-Leibler distance of a topic to a “background” topic, a topic found equally probable in all the documents. Meaningful topics appear in a small subset of the data, thus higher values are preferred.

### 3 Experimental Setting

#### 3.1 Datasets and Preprocessing

To analyze the impact of the length of the documents with respect to several models and performance measures, we consider two different datasets: 20Newsgroup\(^2\) (20NG), where each document is characterized by a long text, and M10 (Lim and Buntine, 2014), which is composed of titles of scientific papers, and therefore it represents a case study of short texts.

We adopt a common preprocessing procedure\(^3\): punctuation removal, lemmatization, removal of English stop-words and unfrequent words, removal of documents with less than 3 words (for M10) or 5 words (for 20NG). The stopwords list is the one provided by MALLET\(^4\). Each dataset is split into training (70%), validation (15%) and test set (15%). Table 1 shows the datasets statistics.

<table>
<thead>
<tr>
<th>Datasets</th>
<th># docs</th>
<th>average # words</th>
<th># unique words</th>
<th># classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>20NG</td>
<td>16309</td>
<td>48</td>
<td>1612</td>
<td>20</td>
</tr>
<tr>
<td>M10</td>
<td>8355</td>
<td>6</td>
<td>1696</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 1: Statistics of the preprocessed datasets.

\(^2\)http://qwone.com/ jason/20Newsgroups/
\(^3\)The preprocessed datasets are already provided by the OCTIS library: https://github.com/mind-Lab/ octis.s
\(^4\)http://mallet.cs.umass.edu/

#### 3.2 Bayesian Optimization and Model Settings

We optimize the models’ hyperparameters using BO for each evaluation metric. We trained each model 30 times and considered the median as the evaluation of the function to be optimized. The initial configurations are randomly sampled via Latin Hypercube Sampling and equal to the number of the hyperparameters to optimize plus 2 (to provide enough configurations for the initial surrogate model). The total number of BO iterations is 30 for LDA and 120 for the other models. We use Random Forests as the surrogate model and the Upper Confidence Bound (UCB) as the acquisition function.

We report the models’ hyperparameters and their corresponding ranges in Table 2.

<table>
<thead>
<tr>
<th>Model/Model Hyperparameter</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA ( \alpha ) prior</td>
<td>([10^{-4}, 10])</td>
</tr>
<tr>
<td>LDA ( \beta ) prior</td>
<td>([10^{-4}, 10])</td>
</tr>
<tr>
<td>Dropout</td>
<td>([0, 1 - 10^{-6}])</td>
</tr>
<tr>
<td>Learning rate</td>
<td>([10^{-6}, 10^{-1}])</td>
</tr>
<tr>
<td>Momentum</td>
<td>([0, 1])</td>
</tr>
<tr>
<td>Activation function</td>
<td>elu, leakyrelu, relu, rrelu, selu, sigmoid, softplus, adaddelta, adagrad, adam, rmsprop, sgd</td>
</tr>
<tr>
<td>NeurLDA/ProdLDA/CTM Optimizer</td>
<td>sgd, adam, rmsprop, sgd</td>
</tr>
<tr>
<td># Neurons</td>
<td>100, 200, ... , 1000</td>
</tr>
<tr>
<td># Layers</td>
<td>1, 2, 3, 4, 5</td>
</tr>
<tr>
<td>Learn priors</td>
<td>true, false</td>
</tr>
<tr>
<td>ETM/ETM-PWE Activation function</td>
<td>elu, leakyrelu, relu, rrelu, selu, softplus, tanh, adaddelta, adagrad, adam, asgd, rmsprop</td>
</tr>
<tr>
<td>Optimizer</td>
<td>sgd, adam, rmsprop, sgd</td>
</tr>
<tr>
<td># Neurons</td>
<td>100, 200, ... , 1000</td>
</tr>
<tr>
<td>ETM Rho size</td>
<td>100, 200, 300</td>
</tr>
</tbody>
</table>

Table 2: Hyperparameters and ranges.

Regarding LDA, we optimize the hyperparameters \( \alpha \) and \( \beta \) priors that the sparsity of the topics in the documents and sparsity of the words in the topic distributions respectively. These hyperparameters are set to range between \( 10^{-4} \) and 10 on a logarithmic scale.

The hyperparameters of the neural models are mainly related to the architecture of the network. For all the neural models, we optimize the dropout...
(ranging between 0 and $1 - 10^{-6}$) and the momentum (ranging between 0 and 1). We optimize the learning rate, that is set to range between $10^{-4}$ and $10^{-1}$, on a logarithm scale. We also consider different variants of activation functions and optimizers.

Regarding NeurLDA, ProdLDA, and CTM in particular, we optimize the number of layers (ranging from 1 to 5), and the number of neurons (ranging from 100 to 1000). For simplicity, each layer has the same number of neurons. Finally, we also consider the hyperparameter learn priors that controls if the priors are learnable parameters.

Following (Bianchi et al., 2021a), we use the contextualized document representations derived from SentenceBERT (Reimers and Gurevych, 2019). We use the pre-trained BERT model fine-tuned on the natural language inference (NLI) task.\(^5\)

Considering ETM and ETM-PWE, in addition to the hyperparameters mentioned above, we only optimize the number of neurons (ranging from 100 to 1000). We follow the original implementation, for which the number of hidden layers is set to 1. For ETM-PWE, we use pre-trained word2vec word embeddings (Mikolov et al., 2013), trained on the Google News corpus (3 million 300-dimension English word vectors).

For the neural models, we set the batch size to 200 and we adopted an early stopping criterion for determining the convergence of each model. We set the remaining model parameters to their default values. To have a fair comparison, we set the number of topics to be discovered equals to the number of classes available in each dataset, i.e. 10 for M10 and 20 for 20NG. For running the experiments, we use the open-source library OCTIS (Terragni et al., 2021a), which already integrates the implementations of the considered models and metrics. It is available at the following link: https://github.com/mind-lab/octis.

### 4 Empirical Analysis and Discussion

#### 4.1 Robustness of Neural Topic Models (RQ1)

In table 3 we report the median of the four evaluation metrics for each topic model obtained by the best hyperparameter configuration. Rows represent the optimized metric (marked as metric*), while columns denote the median of the evaluated metric. The overall best values for each metric and dataset are reported in bold. First of all, we can observe that there is not a model that outperforms the others for all the considered metrics. In fact, it seems that each topic model works better for a specific metric.

In particular, LDA is the topic model that obtains the best performance in terms of KL-B*, thus obtaining topics that are significant rather than background topics. While, the topic models based on the neural variational framework defined in (Srivastava and Sutton, 2017), i.e. NeurLDA, ProdLDA, and CTM, are the ones that obtain the highest diversity. Regarding the topic coherence, CTM obtains the best topic coherence for both datasets. In fact, it improves the performance of ProdLDA (second-best model for the topic coherence) through the incorporation of the contextualized pre-trained representations of the documents. Finally, ETM-PWE outperforms the other models in terms of F1*, probably due to the contribution of the pre-trained word embeddings.

Provided that each topic model seems to reach the best performance only in a specific metric, it follows that they cannot simultaneously guarantee optimal performance for the other metrics. We will further investigate the trade-off between different metrics in Sections 4.2 and 4.3. A complete overview of the best configuration of hyperparameters discovered by BO for all the models and for all the considered evaluation measures is reported in Tables 4, 5, 6, 7, 8 and 9. This would allow a user to choose a promising hyperparameter configuration for the evaluation metric of their interest.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1Score*</th>
<th>IRBO*</th>
<th>NPMI*</th>
<th>KL-B*</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>0.469</td>
<td>0.963</td>
<td>0.064</td>
<td>2.299</td>
</tr>
<tr>
<td>NeurLDA</td>
<td>0.339</td>
<td>1.000</td>
<td>0.067</td>
<td>0.907</td>
</tr>
<tr>
<td>ProdLDA</td>
<td>0.373</td>
<td>0.998</td>
<td>0.107</td>
<td>0.992</td>
</tr>
<tr>
<td>CTM</td>
<td>0.361</td>
<td>0.998</td>
<td>0.118</td>
<td>1.019</td>
</tr>
<tr>
<td>ETM</td>
<td>0.453</td>
<td>0.996</td>
<td>0.080</td>
<td>0.370</td>
</tr>
<tr>
<td>ETM-PWE</td>
<td>0.471</td>
<td>0.986</td>
<td>0.089</td>
<td>0.424</td>
</tr>
</tbody>
</table>

Table 3: Median of each performance metric (columns) for each single-objective optimization (rows).

\(^5\)https://huggingface.co/sentence-transformers/bert-base-nli-mean-tokens
4.2 Impact of the Document Length (RQ2)

We can derive other insights by analyzing Table 3 and comparing the two datasets. In particular, we highlight that for LDA the document length seems to be an invariant when optimizing on the KL-B* metric. This insight can be grasped by considering the KL-B* of LDA (i.e. 2.343 for M10 and 2.299 for 20NG) that, not only are the best performance when compared to the other models, but they suggest that LDA performs well independently on the document length and therefore it guarantees optimal KL-B* both on short and long documents.

Another important insight is about the F1 measures obtained by LDA (0.472 and 0.469), ETM (0.534 and 0.453), and ETM-PWE (0.585 and 0.471), which seem to be not affected by the length of the documents. On the other hand, the results for the F1 measure for NeurLDA, ProdLDA, and CTM (which are based on the same architecture) are affected by the documents’ length, obtaining the best performance for short texts. In these cases, when the models achieve a high F1 on short documents (0.420 by NeurLDA, 0.539 by ProdLDA, and 0.563 by CTM), the performance on short documents is lower (0.339 by NeurLDA, 0.373 by ProdLDA, and 0.361 by CTM).

When optimizing for the IRBO* metric, all the models succeed in obtaining almost completely diverse topics, both for long and short texts. The performance of IRBO* for LDA* is slightly affected when dealing with short texts. Finally, we remark that CTM obtains an excellent topic coherence for both datasets, but, on the other hand, the remaining models seem to be particularly affected when dealing with short texts, assuming NPMI values inferior to 0.

4.3 Metrics-Metrics Correlations (RQ3)

In Figure 1, we report the correlations between the evaluation metrics when a single-objective optimization policy is performed. The rows of the correlation matrices denote the optimized metrics (F1*, IRBO*, KL-B*, and NPMI*), while the columns the non-optimized evaluated measures (F1, IRBO, KL-B, and NPMI). According to these results, we can observe if optimizing a model for a specific metric allows us for an increasing or
<table>
<thead>
<tr>
<th></th>
<th>α prior</th>
<th>β prior</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>20NG</td>
<td>1.332</td>
<td>1.146</td>
<td>0.472</td>
</tr>
<tr>
<td>M10</td>
<td>0.627</td>
<td>1.870</td>
<td>0.469</td>
</tr>
</tbody>
</table>

Table 4: Best configuration of hyperparameters discovered by BO for LDA for each evaluation measure.

<table>
<thead>
<tr>
<th>Activation</th>
<th>Dropout</th>
<th>Learn Priors</th>
<th>Learning Rate</th>
<th>Momentum</th>
<th>Num Layers</th>
<th>Num Neurons</th>
<th>Optimizer</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>20NG</td>
<td>sigmoid</td>
<td>1</td>
<td>0.0997</td>
<td>0.789</td>
<td>1</td>
<td>800</td>
<td>adam</td>
<td>0.373</td>
</tr>
<tr>
<td>M10</td>
<td>elu</td>
<td>1</td>
<td>0.0611</td>
<td>0.742</td>
<td>5</td>
<td>100</td>
<td>adam</td>
<td>0.539</td>
</tr>
</tbody>
</table>

Table 5: Best configuration of hyperparameters discovered by BO for ProdLDA for each evaluation measure.

<table>
<thead>
<tr>
<th>Activation</th>
<th>Dropout</th>
<th>Learn Priors</th>
<th>Learning Rate</th>
<th>Momentum</th>
<th>Num Layers</th>
<th>Num Neurons</th>
<th>Optimizer</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>20NG</td>
<td>sigmoid</td>
<td>0</td>
<td>0.0314</td>
<td>0.575</td>
<td>1</td>
<td>1000</td>
<td>adam</td>
<td>0.339</td>
</tr>
<tr>
<td>M10</td>
<td>elu</td>
<td>0</td>
<td>0.0129</td>
<td>0.756</td>
<td>1</td>
<td>800</td>
<td>rmsprop</td>
<td>0.067</td>
</tr>
</tbody>
</table>

Table 6: Best configuration of hyperparameters discovered by BO for NeurLDA for each evaluation measure.

<table>
<thead>
<tr>
<th>Activation</th>
<th>Dropout</th>
<th>Learn Priors</th>
<th>Learning Rate</th>
<th>Momentum</th>
<th>Num Layers</th>
<th>Num Neurons</th>
<th>Optimizer</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>20NG</td>
<td>leakyrelu</td>
<td>0</td>
<td>0.0018</td>
<td>0.751</td>
<td>1</td>
<td>700</td>
<td>adam</td>
<td>0.361</td>
</tr>
<tr>
<td>M10</td>
<td>leakyrelu</td>
<td>0</td>
<td>0.0097</td>
<td>0.789</td>
<td>1</td>
<td>800</td>
<td>adam</td>
<td>0.998</td>
</tr>
</tbody>
</table>

Table 7: Best configuration of hyperparameters discovered by BO for CTM for each evaluation measure.

<table>
<thead>
<tr>
<th>Activation</th>
<th>BOW norm</th>
<th>Dropout</th>
<th>Learning Rate</th>
<th>Optimizer</th>
<th>Rho</th>
<th>Hidden size</th>
<th>Weight decay</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>20NG</td>
<td>leakyrelu</td>
<td>1</td>
<td>0.315</td>
<td>0.006393</td>
<td>adam</td>
<td>200</td>
<td>800</td>
<td>0.000005</td>
</tr>
<tr>
<td>M10</td>
<td>relu</td>
<td>1</td>
<td>0.058</td>
<td>0.006062</td>
<td>adam</td>
<td>100</td>
<td>600</td>
<td>0.000001</td>
</tr>
</tbody>
</table>

Table 8: Best configuration of hyperparameters discovered by BO for ETM for each evaluation measure.
Table 9: Best configuration of hyperparameters discovered by BO for ETM-PWE for each evaluation measure.

<table>
<thead>
<tr>
<th>Activation</th>
<th>BOW norm</th>
<th>Dropout</th>
<th>Learning Rate</th>
<th>Optimizer</th>
<th>Hidden size</th>
<th>Weight decay</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1*</td>
<td>elu</td>
<td>1</td>
<td>0.814</td>
<td>adam</td>
<td>700</td>
<td>0.000190</td>
<td>0.471</td>
</tr>
<tr>
<td>IRBO*</td>
<td>relu</td>
<td>0</td>
<td>0.918</td>
<td>adam</td>
<td>600</td>
<td>0.001485</td>
<td>0.987</td>
</tr>
<tr>
<td>KL-B*</td>
<td>selu</td>
<td>1</td>
<td>0.157</td>
<td>adam</td>
<td>1000</td>
<td>0.000076</td>
<td>0.424</td>
</tr>
<tr>
<td>NPMI*</td>
<td>elu</td>
<td>0</td>
<td>0.121</td>
<td>rmsprop</td>
<td>1000</td>
<td>0.000004</td>
<td>0.089</td>
</tr>
<tr>
<td>M10 F1*</td>
<td>softplus</td>
<td>0</td>
<td>0.182</td>
<td>adam</td>
<td>800</td>
<td>0.000001</td>
<td>0.585</td>
</tr>
<tr>
<td>IRBO*</td>
<td>selu</td>
<td>1</td>
<td>0.406</td>
<td>adam</td>
<td>1000</td>
<td>0.002974</td>
<td>0.997</td>
</tr>
<tr>
<td>KL-B*</td>
<td>leakyrelu</td>
<td>1</td>
<td>0.051</td>
<td>adam</td>
<td>300</td>
<td>0.000002</td>
<td>0.201</td>
</tr>
<tr>
<td>NPMI*</td>
<td>relu</td>
<td>1</td>
<td>0.500</td>
<td>adam</td>
<td>300</td>
<td>0.000001</td>
<td>-0.070</td>
</tr>
</tbody>
</table>

decreasing performance of the other metrics. In Figure 1, we report the Spearman correlation coefficients between metrics using all the runs of a given experiment.

Concerning LDA, when the model is optimized for the KL-B*, NPMI*, or F1*, then the IRBO is positively correlated with these metrics. It is then sufficient to optimize one of the other metrics to get also diverse topics. This occurs in particular for the KL-B* and NPMI* on long documents (0.87 and 0.98 respectively). It is also interesting to notice that optimizing for KL-B* does not imply a maximization for the F1 and NPMI on long texts. To achieve better topic coherence and classification, we should consider background topics as well.

Focusing on NeurLDA, ProdLDA, and CTM, we do not observe substantial differences between long and short documents. IRBO* is not strongly correlated with the other metrics, especially for long documents. This can be grasped by observing the coefficients IRBO* vs F1, KL-B, and NPMI reported in Figure (1f), (1h) and (1d). On the contrary, optimizing NeurLDA, ProdLDA, and CTM for F1*, NPMI* or KL-B* guarantees, in most of the cases, a good performance on all the metrics both for short and long documents (Figure (1e), (1f), (1g) and (1h)).

Concerning ETM, the difference between long and short documents is clear: the optimization of a given metric can be detrimental to the majority of the other metrics when dealing with short documents. In fact, the optimization of ETM w.r.t. IRBO* and NPMI* originates correlation values with all the other metrics that are close to zero or negative (Figure 1i). On the other hand, F1* and KL-B* seem not to be affected by the difference of the datasets. This suggests that maximizing KL-B* or F1* implies good performance also for other purposes. Focusing on long documents (Figure 1j), the optimization of ETM w.r.t. F1*, KL-B*, and NPMI* originates positive correlation values for all the other metrics. On the other hand, we can highlight that optimizing the topic diversity IRBO* does not allow us to simultaneously obtain good performance on topic coherence (NPMI) on long documents. Regarding ETM-PWE, we do not notice a clear difference between the two datasets. The introduction of the pre-trained word embedding into the training process of the model seems to be beneficial for all the metrics.

To summarize, optimizing the neural models according to the IRBO* is not always convenient and may lead to incoherent topics or poor document classification performance. Another important insight concerns the optimization of F1*, which usually guarantees to maximize IRBO, KL-B, and NPMI, for both short and long documents, except for LDA.

5 Conclusions and Future Work

In this paper, we presented an empirical analysis of Neural Topic Models to determine the relationship between hyperparameters, document length and performance measures. Three main research questions have been addressed for understanding under which conditions such Topic Models could work for guaranteeing their best performance.

The main findings could help both practitioners on tuning the models according to their objectives and researchers to explore the role of hyperparameters and document length with respect to any given performance measure. Regarding the future work, the problem of hyperparameter optimization by considering multi-objective optimization (Horn and Bischl, 2016) will be addressed for understanding to which extent multiple metrics could be optimized according to the length of the documents and the hyperparameters of the models.
References


TR-SEQ: Named Entity Recognition Dataset for Turkish Search Engine Queries

Berkay Topcu
AI Industry Solutions
Turkcell Technology
İstanbul, Turkey
berkay.topcu@turkcell.com.tr

İlknur Durgar El-Kahlout
NLP Technologies
Turkcell Technology
İstanbul, Turkey
ilknur.durgar@turkcell.com.tr

Abstract

Recognizing named entities in short search engine queries is a difficult task due to their weaker contextual information compared to long sentences. Standard named entity recognition (NER) systems that are trained on grammatically correct and long sentences fail to perform well on such queries. In this study, we share our efforts towards creating a cleaned and labeled dataset of real Turkish search engine queries (TR-SEQ) and introduce an extended label set to satisfy the search engine needs. A NER system is trained by applying the state-of-the-art deep learning model BERT to the collected data and its high performance on search engine queries is reported. Moreover, we compare our results with the state-of-the-art Turkish NER systems.

1 Introduction

Named entity recognition (NER) aims to identify the predetermined entity categories (person, organization, location, etc.) accurately in a given text. NER studies which is one of the most widely studied subjects in the field of natural language processing (NLP), has been started for English and spread over a range of languages including Turkish (Yeniterzi, 2011). Initial NER studies for Turkish used small labeled datasets that contain grammatically correct and typo-free sentences. Since manual labeling is a time-consuming and a costly process, gathering larger datasets and manually labeling them has not been feasible. With the increase of internet use, correctly classifying named entities has become a crucial need especially for social media entries and search engine (SE) queries. In search engines, identifying entities such as famous persons, places or dates in user queries is crucial in order to accurately determine the information to be displayed to the user and response with the right information. For example, if the user is searching for a famous person, SE should return the Wikipedia information about the person. Moreover if this famous person is an artist, the identified named entity assists in conveying the details of his/her movie/album/artwork details. Similarly, it is important to determine the location in the query for which we want to know the weather.

The state-of-the-art Turkish NER tools fail to perform well on search engine queries due to two main reasons. The first reason is that most of the NER tools that have been developed for Turkish so far generally carry out categorization studies with three classes (Person, Organization and Location). Naturally, a three-class model cannot detect all the named entities in a SE query that is required to create a successful SE response. For this reason, it has become a necessity to determine the entity types specific to the SEs and to make a special development for these types. The second reason is that NER tools are dependent on the texts and domains they are trained on. NER tools, which are usually trained in canonical and grammatically correct long sentences, do not succeed in short SE queries that contain abbreviations and spelling errors and mostly consist of two or three words. As the short inputs have poor contextual knowledge compared to long sentences and often include misspellings, existing NER tools are observed to perform poorly on these type of texts and the success rates are far below the desired levels.

Although the first attempts for Turkish NER systems are dominated by rule-based systems (Kıčük and Yazıcı, 2009), CRF-based models (Yeniterzi, 2011; Seker and Eryiğit, 2012; Yeniterzi et al., 2018) came to the fore in the forthcoming studies. In all of these studies, a relatively small dataset consisting of about 30K sentences, clean in terms of spelling and grammar, is used (Tur et al., 2003) and models are based on three classes (person, location, organization). Recently, models using deep learning methods (Demir and Ozgur, 2014; Aras et al., 2021), especially Bidirectional Long Short-Term
Memory (BLSTM), became more popular (Güneş and Tantuğ, 2018).

Even though the accuracies of these NER studies are high in their test sets, the performance drop is dramatical on social media (e.g., Twitter) and SE queries (Çelikkaya et al., 2013; Küçük and Steinberger, 2014). For this reason, studies are carried out to collect dedicated data for social media texts (Okur et al., 2016). The most recent studies in Turkish NER field utilize Transformers and BERT based models (Yıldırım, 2019; Akdemir and Güngör, 2019; Aras et al., 2021).

In this paper, we focus on the recognition of a selected group of named entities for Turkish search engine queries. In the study, Yaani\(^1\) search engine entries are collected, the categories for the SE needs are determined and a labeling study is carried out using an in-house annotation tool. The data consisting of a total of 100\(K\) SE queries is labeled by more than one person and a verification study is carried out with the obtained outputs. In addition, a high-performance NER tool is developed by training the Transformers-based BERT (Devlin et al., 2018) model, which is one of the most successful approaches and widely used in current NLP studies, with our labeled dataset, and an accuracy of 90.41% is achieved.\(^2\)

2 Dataset Creation Process

For data-driven NER systems to perform well on real world scenarios, it is of crucial importance to have training data that has a similar distribution with the real world data. In order to satisfy this constraint, we carried out an extensive data collection and labeling process for a NER system specific to SE queries. We collected a dataset of 100\(K\) queries that are submitted to Yaani search engine for a specific period of time without having a constraint related to short/long queries. Next, we cleaned the malicious content and normalized the spelling errors manually. Finally, we removed the duplicates which resulted in a dataset with \(97,428\) queries.

2.1 Determining Entity Classes

Active usage of widgets and snippets is a particular way of enhancing user experience in search engines. Currently, most of the well-known search engines respond to user queries via widgets and snippets which convey information requested by the user in a concise and compact manner. The main motivation behind this study is the extraction of relevant information obtained from user queries and activate related widgets and snippets accordingly. At the time of this study, recognition of named entities specific to particular widgets such as weather, currency converter, maps and info boxes (i.e., Wikipedia results) have been focused.

Due to the inadequacy of the standard three-class model for a SE specific NER model; we examined 48 different entity categories provided by Shrinked TWNERTC Turkish NER Data \(^3\) (Şahin et al., 2017) and selected the following seven main categories to cover different agents such as weather, maps, and currency inquiries. These classes are as follows;

- **Person**: All type of proper names for persons
  Ex: Mustafa Kemal Atatürk, Prof. Dr. Ali Ünal

- **Organization**: All type of entities that have an organization or that are organized such as government institutions, firms, hospitals, universities, soccer clubs, festivals Ex: Walter Reed Askeri Hastanesi (Walter Reed Military Hospital), İstanbul Üniversitesi (İstanbul University)

- **Location**: All places that have a physical position like countries, cities, villages, etc. Ex: İstanbul, Van Gölü (Van Lake), Hollywood

- **Date**: All date/time entries plus important dates like "mother’s day" etc. Ex: 15.02.2020, 15 Ekim (November, 15th), M.Ö 500 (500 B.C.), 1870

- **Measure**: All entities that are in the global standard, the International System of Units (SI), such as, kg, lt, etc. Ex: 1700 metrekare (square meters), 2 lt (liters), 300 metre (meters)

- **Currency**: All currency entities plus commodities such as gold, silver and cryptocurrency Ex: 19 milyon TL (milion Turkish Liras), 20 bitcoin, 10 gram altın (gram of gold)

- **Production_art_music**: All entities that are produced like films, series, songs, books, etc.

\(^1\)yaani.com.tr
\(^2\)The data is freely available for academic purposes. Please contact the authors for dataset acquisition.
2.2 Labeling Process

Following the category determination, we organized a team of twenty people for data labeling. We shared a labeling guide in which we itemized the tagging criteria and provided positive and negative examples. We also asked annotators to complete a demo task before initializing the main project. By the help of the demo outputs, we revised the labeling guide and finalized the rule set as follows:

- A NE should be labelled to include the longest phrase that can be retrieved. **Ex:** Bursa Büyükşehir Belediyesi (Bursa Metropolitan Municipality) should be labeled as **Organization** as a whole instead of labeling only Bursa as **Location**.

- Successive NEs of the same type should be labeled individually if separated by "," or "and". **Ex:** Washington DC White House should be labelled seperately in Washington DC'deki Beyaz Saray'a nasıl giderim (How can I go to the White House in Washington DC), instead of selecting it as a whole phrase such as White House in Washington DC.

- While labeling person NEs, if the title phrase does not have another NE, the whole title will be labeled as **Person**. **Ex:** İçleri Bakamı (Minister of Interior) Soylu will be labeled as a whole as **Person** but as the query Bursa Büyükşehir Belediyesi Başkanı (Mayor of Bursa Metropolitan Municipality) Alinur Aktaş has an **Organization** NE in the phrase Bursa Büyükşehir Belediyesi (Bursa Metropolitan Municipality), labeller will seperately label this phrase and only label Alinur Aktaş as **Person**.

- General names following a proper name should not be included in the NE except the location names including "Lake", "Street" and "University" etc. **Ex:** İstanbul should be labeled as **Location** in the query İstanbul ilinde hava durumu (Weather forecast in Istanbul city), on the other side, location names such as Van Gölü (Van Lake), İstiklal Caddesi (İstiklal Street) should be labeled with the following location name.

- Due to the Turkish morphology, an apostrophe is attached at the end of the proper name if it is followed by a suffix. Such words with apostrophes should be labeled as a whole. **Ex:** Yunanistan’ın (Greece’s) should be labeled instead of Yunanistan (Greece) in queries.

- Subordinates consisting more than two or more words should be labeled as a whole.

During the whole process, we employed an in-house labeling and verification tool. Figure 1 shows an example annotation screen used in the study. In the light of the given instructions, the annotator labeled the phrase Fazilet Hanım ve Kızları (a Turkish serie) as **Production_art_music** and Ege- menler Yalısı (a location in this serie) as **Location**.

### 2.3 Consolidation Step: Annotator Disagreement

The first three weeks of the annotation process was dedicated to the labeling of the queries and the final week was used to check the mismatches between
annotators and consolidate the output. Almost half of the queries (45,442) are labeled by two different annotators. These queries are categorized into three groups; i) queries that both annotators agree on, ii) queries that two annotators do not agree on, and iii) queries that one of the annotators labels some words and the other annotator does not label any of the words.

Leaving out the queries that both annotators agree on, 10,800 queries are considered in the consolidation step in which four experts (different from the annotators) checked the mismatching labels. If any of the labels is correct, it is selected as the final label. If both labels are wrong, the true label is determined by the expert. Figure 2 illustrates the consolidation screen used in order to analyze the mismatch between the annotations of a query. Regarding the rule that the longest phrase should be retrieved, the label from the first annotator is kept and the label from the second annotator is discarded.

2.4 Dataset Statistics

The collected dataset consists of 97,428 search engine queries and includes 294,100 words. Figure 3 illustrates the distribution of query lengths. The dataset mostly contains short word sequences of length between one and four. The average length of the queries is 3.018. The longest query has 35 words whereas the shortest one is a single word.

Table 1 provides the number of different named entity types used in this study and the percentage of multi-word entities for each type. The most frequent named entity type is Organization. Person, Production_art_music and Location are three other named entity types that occur too often. Measure is the rarest type whereas Date and Currency are not observed very frequently.

3 Experiments and Results

Using this new dataset, we trained a Turkish NER system and demonstrated the improved NER performance of using short search queries during training. Our Turkish NER system is trained by fine-tuning the BERTurk (Schweter, 2020) model using our dataset. BERTurk is a community driven BERT (Bidirectional Encoder Representations for Trans-
<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC-BERT</td>
<td>0.9041</td>
<td>0.7302</td>
<td>0.8131</td>
<td>0.7695</td>
</tr>
<tr>
<td>TC-ELECTRA</td>
<td>0.8982</td>
<td>0.7056</td>
<td>0.8077</td>
<td>0.7532</td>
</tr>
</tbody>
</table>

Table 2: Seven class NER performance

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turkish-Bert-NLP (Yıldırım, 2019)</td>
<td>0.5055</td>
<td>0.2007</td>
<td>0.5552</td>
<td>0.2948</td>
</tr>
<tr>
<td>Char-BiLSTM-CRF (Aras et al., 2021)</td>
<td>0.8649</td>
<td>0.4968</td>
<td>0.3951</td>
<td>0.4402</td>
</tr>
<tr>
<td>BERTurk-CRF (Aras et al., 2021)</td>
<td>0.8804</td>
<td>0.5844</td>
<td>0.4843</td>
<td>0.5296</td>
</tr>
<tr>
<td>TC-BERT</td>
<td>0.9457</td>
<td>0.7491</td>
<td>0.8561</td>
<td>0.7991</td>
</tr>
</tbody>
</table>

Table 3: Comparison of three class NER performances

formers) (Devlin et al., 2018) model for Turkish. Its training corpus has a size of 35GB and 44B tokens with a vocabulary size of 128K. For fine-tuning the BERTurk model for NE classification, \( \text{max} \_\text{seq} \_\text{length} \) is set to 128, \( \text{train} \_\text{batch} \_\text{size} \) and \( \text{eval} \_\text{batch} \_\text{size} \) are set to 32, \( \text{learning} \_\text{rate} \) is set to \( 2e^{-5} \) and the model is trained for 5 epochs.

In order to evaluate the performance of the NER system, the dataset is split into three: train, validation and test. For the test set, 1% of the data is selected randomly from the whole dataset by preserving the distribution of named entity types. Similarly, 1% is randomly selected as validation data and the remaining 98% is used for training. Using 1% of the whole dataset for testing purposes resulted about 1000 test queries and all seven classes are included proportionally to their distribution in the dataset.

In addition to the BERT model, we have also experimented with a pre-trained version of ELECTRA (Clark et al., 2020) model (also publicly available within the BERTurk study), which allowed us to compare the performance of the collected dataset on different deep architectures. Table 2 illustrates the performance of the NER systems (TC-BERT and TC-ELECTRA) fine-tuned using our dataset.

Publicly available Turkish NER models and other studies in the literature usually take three named entity types (Person, Organization and Location) into account. The effort towards increasing the number of named entity types is very limited for Turkish and to the best of our knowledge there is no other study that focuses on the named entity types discussed in this paper. Therefore, comparing the performances of our system with previous studies on the same test set is not immediately clear.

For a fair comparison, we removed the remaining labels from our test set by keeping three base named entity types (Person, Organization and Location). Since our model is based on seven named entity types, we filtered out the output labels that belong to other types (by masking as Other) and generated the final results. We compare our results with two recent Turkish NER studies: i) the NER model of “Turkish-Bert-NLP-Pipeline” (Yıldırım, 2019) that is known for its high performance on well-formed Turkish texts and ii) recent neural sequence tagging models discussed in (Aras et al., 2021). As depicted in Table 3, the NER system trained using our dataset of short queries outperforms the systems trained using grammarly correct Turkish sentences, both in terms of accuracy and F-measure.

4 Conclusion

This study presents a new Turkish NER dataset which is created specifically for short word sequences that comprises a large portion of the search engine queries. Our aim is to improve the NER performance of the models that are trained using grammatically correct sentences and fail to perform well on search engine queries. The dataset is created by cleaning and labeling 100K search queries and provides a rich resource for Turkish NER studies. In addition, our transformer based NER system presents useful baseline accuracies for future studies.

Our next step is to train a BERT-based language model from scratch by using search engine queries and replace the pre-trained BERT model provided by BERTurk. This will allow us to develop a NER system specific to short queries. Furthermore, creating a knowledge graph by utilizing the named entity labels will be an important step in enhancing
overall search engine performance.

References


Opinion Prediction with User Fingerprinting

Kishore Tumarada¹, Dr. Yifan Zhang¹, Dr. Fan Yang², Dr. Eduard Dragut³, Dr. Omprakash Gnawali¹, and Dr. Arjun Mukherjee¹

¹Department of Computer science, University of Houston, USA
²Amazon, USA {The paper was done before the author joined Amazon.}
³Department of Computer and information systems, Temple University, USA
{kishore.t04, aeryen, fyangskip}@gmail.com, edragut@temple.edu, {odgnawal, amukher6}@central.uh.edu

Abstract

Opinion prediction is an emerging research area with diverse real-world applications, such as market research and situational awareness. We identify two lines of approaches to the problem of opinion prediction. One uses topic-based sentiment analysis with time-series modeling, while the latter uses static embedding of text. The latter approaches seek user-specific solutions by generating user fingerprints. Such approaches are useful in predicting user’s reactions to unseen content. In this work, we propose a novel dynamic fingerprinting method that leverages contextual embedding of user’s comments conditioned on relevant user’s reading history. We integrate BERT variants with a recurrent neural network to generate predictions. The results show up to 13% improvement in micro F1-score compared to previous approaches. Experimental results show novel insights that were previously unknown such as better predictions for an increase in dynamic history length, the impact of the nature of the article on performance, thereby laying the foundation for further research.

1 Introduction

Sentiment analysis plays a key role in economic, social, and political contexts. Companies can understand customer’s opinions based on reviews and/or social media conversations to make fast and accurate product decisions. They can index unstructured customer data at scale based on broad sentiments such as positive, negative, or neutral. It also enables market research - explore new markets, anticipate future trends, and seek an edge on the competition. In a political context, sentiment analysis can be useful in understanding political homophily using tweets analysis (Caetano et al., 2018).

Moreover, as social media has emerged as a new source of communication, governments can analyze people’s reactions to issues such as police-related encounters, mob protests and anticipate responses before they turn violent. In this context, predicting people’s opinion before expressing is an important next step in applying sentiment analysis to real-world applications. We are approaching this problem through sentiment prediction, especially on current news events, that can raise situational awareness, understand the future viewpoints of the citizenry on pressing social and political issues.

Recently fine-grained models generated a person’s fingerprint, based on one’s recent reading and response history, to predict response on an unknown event (Yang et al., 2020). In this work, we propose a novel architecture to create a dynamic fingerprint of a user that is contingent upon the target event. We choose to evaluate our models on the dataset used by Yang et al., 2020, which contains newspaper articles as events and users’ comments on them as opinions. Those ground truth opinions are used as a basis for sentiment prediction in this work.

Our model consists of three main steps. In the first step, relevant articles are extracted from a user’s reading history based on their similarity to target article. In the second step, the contextual embedding of these relevant articles, conditioned on the target article, is used to create a reading track. Similarly, we create a writing (response) track with the contextual embedding of extracted articles with corresponding comments by the user. Lastly, dynamic fingerprints are generated based on the temporal pattern of the reading and writing tracks. These dynamic fingerprint vectors for a particular user are then used to predict the user’s sentiment on the target event.

Our main contributions in this paper are:

1. A novel architecture of dynamic fingerprint generation based on the contextual embedding of the user’s reading history.
2. Experimental results show that our method
outperformed the previous approach over various news outlets datasets.

2 Related work

Opinion prediction based on the temporal pattern of sentiments is a relatively new research topic, but basic concepts such as sentiment analysis, question answering based on dialogue context have been explored in different communities and settings.

Si et al., 2013 proposed a technique to leverage topic-based Twitter sentiments to predict the stock market using vector autoregression and Dirichlet process mixture models. Li et al., 2019 proposed a time+user dual attention-based LSTM network to perform emotional analysis on Chinese public opinion texts on social networking platform. But they did not use contextual embedding and explore the prospect of generating a unique user fingerprint before predicting sentiment.

Conversational question answering (CQA) is an emerging research area in the machine reading comprehension task (MRC). For single-turn MRC tasks, contextualized language representation using BERT has obtained state-of-the-art scores on SQuAD datasets (Devlin et al., 2019). CQA is a multi-turn question answering task that includes passage comprehension, contextual understanding, and coreference resolution. Zhu et al. have proposed SDNet (Zhu et al., 2018) to solve this problem by concatenating previous questions and answers as one query to fuse context into traditional MRC models by leveraging BERT, attention, and RNN techniques. Similarly, Ohsugi et al., 2019 have proposed fine-tuning approach with BERT in a multi-turn context by modeling the interaction between paragraph and dialogue history independently.

However, these models cannot be applied to the present problem since they did not integrate the concepts of the sequential pattern of sentiments along with the unique fingerprint of each user, which can play a key role in predicting the future opinion of a user on different topics.

3 Proposed model and methodology

In this section, we propose two classes of FingerPrint Embedding models (FPE) - Static and Dynamic - for the task of predicting the sentiment of a user $u$ to a new article. In a narrow sense, we used the term static to refer to the approach of using recent comment history, which is independent of the nature of target article $A(t)$, and dynamic to refer to the approach of using articles relevant to target article $A(t)$ in the overall history of user’s comments, which are dependent on the nature of target article. In a broad sense, static and dynamic terms distinguish the way target article $A(t)$ is integrated with user’s reading history to generate the fingerprint.

We used the user’s commenting history on articles that they read. We assume that we know the sentiment of each comment. (This can be obtained with one of the many sentiment analysis tools.) Formally, we are given the articles, comments along with the sentiments of a user $u$, i.e., $(A_1, C_1, S_1), (A_1, C_2, S_2), ..., (A_2, C_j, S_j), ..., (A_{t-1}, C_n, S_n)$, and the goal is to predict the sentiment $S_t$ of $u$’s response to unseen article $A_t$. In general, $n > t$ because a user may post multiple comments to an article.

The overall architecture includes history selection, text embedding, fingerprint creation, and lastly sentiment prediction. We describe them below.

3.1 History selection

We have explored two methods of history selection - static history and dynamic history.

Given an article $A_t$, its static history, according to the user $u$, is the list of the most recent article-comment pairs posted by $u$. Depending on the magnitude of $s$ and number of comments made by the user, the list may include comments from one article or multiple articles. We use this method in the Static FPE model.

For the dynamic history of article $A_t$, we have ranked all articles of author’s reading history based on similarity and picked top $r$ articles, along with their comments, as shown in Figure 1. Here similarity between articles is calculated using DistilRoBERTa Semantic textual similarity model (Reimers and Gurevych, 2019).

It is a variant of sentence-BERT, a modified pre-trained BERT network that uses siamese and triplet network structures to derive semantically meaningful sentence embeddings that are then compared using the cosine-similarity metric.

Table 1 shows the article samples extracted by both methods. For the target article “Pelosi says house will condemn all hate as anti-semitism debate overshadows congress”, we can see that articles in the dynamic history, especially articles 1,3,4
Target article: Pelosi Says House Will Condemn All Hate as Anti-Semitism Debate Overshadows Congress

<table>
<thead>
<tr>
<th>S No.</th>
<th>Static history</th>
<th>Dynamic history</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ilhan Omar Knows Exactly What She Is Doing</td>
<td>With Control of Congress at Stake, Trump Rephrases a Favorite Theme: Fear Immigrants</td>
</tr>
<tr>
<td>2</td>
<td>Ilhan Omar Controversy Caps a Month of Stumbles for Democratic Leaders</td>
<td>After Loss of House, Trump Makes Overture to Democrats, Coupled With Threats</td>
</tr>
<tr>
<td>4</td>
<td>Pelosi Says House Will Condemn All Hate as Anti-Semitism Debate Overshadows Congress</td>
<td>White House Considers Using Storm Aid Funds as a Way to Pay for the Border Wall</td>
</tr>
<tr>
<td>5</td>
<td>Tariff Man Has Become Deficit Man</td>
<td>Senate Leaders Plan Competing Bills to End Shutdown</td>
</tr>
<tr>
<td>6</td>
<td>Paul Manafort to Be Sentenced Thursday in 1 of 2 Cases Against Him</td>
<td>House Votes to Block Trump’s National Emergency Declaration About the Border</td>
</tr>
</tbody>
</table>

Table 1: Articles extracted by different methods of history selection. We can see that dynamic history of articles are more relevant than static history.

are more related to the themes such as congress, hate etc., of target article. But static history does not show any relevant articles, except for article 4, which was also because the user has read the target article in the recent past. So we can see that dynamic history has a more pertinent set of articles than static history.

3.2 Text embedding

In this stage, we create reading and writing tracks based on the selected user’s history. Specifically, we encode articles and comments using contextual embedding by two types of encoders (BERT variants) - DistilBERT (Sanh et al., 2019) and ELECTRAN (Clark et al., 2020) models. However, the proposed models are open to any variant of BERT encoder. We have used encoders in two modes - single-sentence mode where single span of contiguous text is encoded in form a special classification token \([CLS]\); two-sentence mode where two spans of text separated by \([SEP]\) token are encoded as \([CLS]\).

In the static FPE model, as shown in Figure 2, reading track comprises of selected articles, encoded by fixed-length \([CLS]\) tokens of single-sentence mode of BERT variant models, i.e., \([CLS]_{t-1}, ..., CLS_{t-2}, CLS_{t-1}\). We have conducted experiments both on - pretrained (with frozen parameters) and trained - BERT variants. Writing track comprises of both article and comments at each time step \(k [A_k, C_k]\), encoded as \([CLS]\) token outputs of two-sentence mode of BERT variant models, similar to reading track, i.e.,
For the Static FPE model, in stage 1, for a target article \( A(t) \), \( s \) article-comment pairs from \((t-s)\) to \((t-1)\) immediate past from a particular user’s history are selected. In stage 2, the contextual embedding of \( s \) articles are generated, and corresponding [CLS] tokens from DistilBERT/ELECTRA models are extracted as reading track \( R(t-s) \) to \( R(t-1) \). Similarly, contextual embedding of articles \( A(t-s) \) to \( A(t-1) \) conditioned with corresponding comments \( C(t-s) \) to \( C(t-1) \) are generated and [CLS] tokens are extracted as writing track \( W(t-s) \) to \( W(t-1) \). In stage 3, an RNN with 2 hidden layers is trained with concatenation of both tracks against corresponding comment sentiment \( S(t) \) at each time step \( t \) and the last hidden layer output is extracted as the fingerprint of user. In the last stage, a Multi-layer perceptron (MLP) is trained on the concatenation of the fingerprint \( h^f_t \) and [CLS] against the sentiment \( S(t) \) of the user’s comment on article \( A(t) \).

\[
[CLS]_{t-s}, ..., CLS_{t-2}^W, CLS_{t-1}^W.
\]

In the dynamic FPE model, as shown in Figure 3, for the reading track - target article \( A_t \) is appended to every article of relevant history and both of them are encoded by two-sentence mode of BERT variant models as [CLS] token outputs, i.e., \([CLS]_{t-r}, ..., CLS_{t-2}^R, CLS_{t-1}^R\). The writing track is similar to that of static FPE model, i.e., \([CLS]_{t-r}^W, ..., CLS_{t-2}^W, CLS_{t-1}^W\).

Lastly, both the tracks are concatenated at each time step to create a unified fingerprint in case of static FPE model. But in dynamic FPE model, these tracks are used as separate entities to create two different fingerprints in the next stage.

### 3.3 Fingerprint creation

In this stage, unidirectional RNN is used to form contextual understanding of both reading and writing tracks.

In the static FPE model, each [CLS] token of reading track is concatenated with corresponding [CLS] token of writing track at each time step, i.e., \([CLS]_{t-r}^R, CLS_{t-s}^W, ..., CLS_{t-2}^R, CLS_{t-2}^W, CLS_{t-1}^R, CLS_{t-1}^W\). We have used Gated Recurrent Unit (GRU) (Chung et al., 2014) instead of LSTM, because the former has fewer parameters, trains faster with comparable performance to the latter. A two-layer GRU network is trained with the above concatenated output against corresponding sentiments \( S_{t-s}, ..., S_{t-2}, S_{t-1} \). The last hidden layer output is taken as the fingerprint embedding \( h^f_t \) for article \( A_t \) of user \( u \).

In the Dynamic FPE model, we create a separate fingerprint for the reading track and writing track, respectively, with 2-layer RNN and 1-layer RNN networks. Both are GRU networks. An additional layer is used for the reading track to get a more complex feature representation of the relationship between target articles and the relevant history of articles. Here, fingerprint embedding is the concatenation of the last hidden layer outputs of both networks, i.e., \([h_t^R; h_t^W]\).

### 3.4 Sentiment prediction

Lastly, a one-layer Multi-Layer Perceptron (MLP) is trained with concatenation of fingerprint embedding and [CLS] token embedding of the target article, as input against the sentiment of response. In the static FPE model, the MLP is trained on \([h^f_t; CLS^A_t]\) against output \( S_t \). Whereas in the dynamic FPE model, it is trained on \([h_t^R; h_t^W; CLS^A_t]\) against output \( S_t \).

## 4 Experiments

The main goals of our experiments are:

1. Measure the performance (in terms of micro
Figure 3: For the Dynamic FPE model, in stage 1, for a target article \( A(t) \), \( r \) relevant article-comment pairs from a particular user’s complete history are selected based on semantic textual similarity. In stage 2, the contextual embedding of each of the \( r \) articles conditioned with target article \( A(t) \) is generated and corresponding [CLS] tokens from DistilBERT/ELECTRA models are extracted as reading track. Similarly, the contextual embedding of articles \( A(t-r) \) to \( A(t-1) \) conditioned with corresponding comments \( C(t-r) \) to \( C(t-1) \) are generated, and [CLS] tokens from DistilBERT/ELECTRA models are extracted as writing track. In stage 3, the reading track is encoded with a two-layered RNN trained against the corresponding comment sentiment \( S(t-r) \) to \( S(t-1) \) at each time step and the last hidden layer output is extracted as the reading fingerprint of the user. Similarly, the writing track is also encoded with a one-layered RNN and the last hidden layer output is extracted as a writing fingerprint. In the last stage, an MLP is trained on the concatenation of the reading fingerprint \( h_f^R \), writing fingerprint \( h_f^W \) and [CLS] \( A(t) \) against sentiment \( S(t) \) of corresponding comment \( C(t) \) of user on article \( A(t) \).

F1-score) of both static and dynamic model variants in the prediction of the sentiment of a user to an unknown article

2. Analysis of model performance by studying the impact of dynamic history length and nature of articles on prediction.

4.1 Data preparation

We perform our empirical study on the datasets used for Personal opinion prediction by Yang et al., 2020. In these datasets, news articles are randomly selected from Archiveis, The Guardian, and New York Times. We do not consider users with fewer than ten comments. If after this step an article remains without any user, the article is discarded. We checked manually a random subset of articles and their comments and found that irrelevant comments are very few to ignore.

Each input example case comprises a target article/comment and its corresponding selected history of article-comment pairs. For each user, the data is split into training and test sets in the ratio of 90:10 ratio sequentially, i.e., the last 10% of comments made by user chronologically as test data and the remaining as training data. Also, the training set is split into training and validation data in the ratio of 90:10 preserving the sequential order.

Since our task is to predict the sentiment (as a score in \([-1, 1]\)) on a future comment, we considered 4 models to conceive the sentiment score. They are - Vader (Hutto and Gilbert, 2014), Flair (Akbik et al., 2019), BlobText sentiment, and BlobText subjectivity (Loria et al., 2014), to automatically label all comments. We assume that users have consistent views and stances on the same event within these articles and comments. Vader is a rule-based model for general sentiment analysis. It is constructed from a generalizable, valence-based, human-curated gold standard sentiment lexicon. When assessing the sentiment of tweets, Vader outperforms individual human raters (Hutto and Gilbert, 2014). Flair presents a unified interface for all word embeddings and supports methods for producing vector representation of entire documents. We use the Flair pre-trained classification model for sentiment labels. The model is trained on the IMDB dataset and has 90.54 micro F1-score. BlobText is a simple rule-based API for sentiment analysis. It has both sentiment model and subjectivity model, and we refer to them as Bsent and Bsubj respectively.

We used article titles and comment content as the basis for our model. We did not use article content since they are extremely long and moreover our focus is on user’s opinion on the article, not the article per se. Table 2 shows the dataset statistics.
Figure 4: Micro F1-scores of all models on test data for all outlets with baseline scores, shown as white empty circle in the figure. We can see that dynamic FPE models have better performance of 4-13% points over baseline FPE model, except for Vader sentiment, across all three outlets.

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Archives</th>
<th>TheGuardian</th>
<th>NewYorkTimes</th>
</tr>
</thead>
<tbody>
<tr>
<td># U</td>
<td>20,920</td>
<td>41,069</td>
<td>37,957</td>
</tr>
<tr>
<td># Mean C_u</td>
<td>25.8</td>
<td>33.63</td>
<td>32.26</td>
</tr>
<tr>
<td># Med. C_u</td>
<td>9</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td># A</td>
<td>2,043</td>
<td>6,393</td>
<td>3,647</td>
</tr>
<tr>
<td># C</td>
<td>812,768</td>
<td>5,467,755</td>
<td>2,328,597</td>
</tr>
</tbody>
</table>

Table 2: Statistics for three news datasets. For each dataset, # U refers to total number of users, # mean C_u - mean number of comments per user, # med. C_u - median number of comments per user, # A - total number of articles and # C - total number of comments.

4.2 Baselines and experiment settings

We evaluated our models against the baseline Fingerprint embedding (FPE) model (Yang et al., 2020). In FPE, recent history of a target article was extracted and then Byte-Pair Embedding (BPE) (Heinzerling and Strube, 2018) and GRU were used to encode the words in articles and comments into fixed-length vectors. Subsequently, user’s fingerprint was generated using a second GRU that modeled the sequence of history, which was a direct concatenation of prior articles and comments, encoded as fixed-length vectors. Finally, the concatenation of fingerprint embedding and target article embedding was given to MLP to predict the sentiment.

On the contrary, we examined both recent and relevant history of target article. We also used BERT based contextual embedding to encode the relationship between articles and comments rather than separate encoding. Finally, we created fingerprint separately for reading and writing track and then concatenated in the final stage for predicting the sentiment.

4.3 Implementation

As discussed in Section 3, we have primarily two models - static FPE and dynamic FPE- with different variants of each by using two types of contextual encoders- DistilBERT and ELECTRA, along with frozen and trained parameters. For static FPE models, we have taken an arbitrary history length of 12 article-comment pairs, so $s = 12$ for fingerprinting. For dynamic FPE models, we have taken relevant dynamic history length of 15 article-comment pairs, $r = 15$, after comparing micro F1-scores for various lengths from 5 to 20. We discuss about the impact of dynamic history length on performance in Section 5. In all cases, a hidden layer of GRU with a dimension of 256 is created. We have trained all model variants for 10 epochs and saved the model based on the mean micro F1-score over all four sentiments on the validation dataset. Usually, the best model is achieved around 5-6 epochs.
Model Type BERT (parameters) history type history length

<table>
<thead>
<tr>
<th>Model</th>
<th>Type</th>
<th>BERT variant</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>sFPE-BERT-frozen</td>
<td>static</td>
<td>DistilBERT static</td>
<td>frozen</td>
<td>12</td>
</tr>
<tr>
<td>sFPE-BERT-trained</td>
<td>static</td>
<td>DistilBERT static</td>
<td>trained</td>
<td>12</td>
</tr>
<tr>
<td>sFPE-ELEC-frozen</td>
<td>static</td>
<td>ELECTRA static</td>
<td>frozen</td>
<td>12</td>
</tr>
<tr>
<td>sFPE-ELEC-trained</td>
<td>static</td>
<td>ELECTRA static</td>
<td>trained</td>
<td>12</td>
</tr>
<tr>
<td>dFPE-ELEC-frozen-dhist</td>
<td>dynamic</td>
<td>ELECTRA dynamic</td>
<td>frozen</td>
<td>15</td>
</tr>
<tr>
<td>dFPE-ELEC-frozen-dhist</td>
<td>dynamic</td>
<td>DistilBERT dynamic</td>
<td>frozen</td>
<td>15</td>
</tr>
</tbody>
</table>

We use Adam optimizer with weight decay (Loshchilov and Hutter, 2019) and a schedule of learning rate (lr) that decreases following the values of the cosine function between the initial lr set in the optimizer to 0, with several hard restarts, after a warmup period during which it increases linearly between 0 and the initial lr set to 0.001 in the optimizer. We implemented the model using PyTorch lightning, a wrapper for PyTorch. The code is released at GitHub.

5 Results and discussion

Table 3 describes the model notation and configuration used in the results. Figure 4 shows that except for vader sentiment, all our model variants outperform the FPE baseline. In the remaining three sentiments, dynamic-FPE-ELECTRA (frozen) model with either dynamic or static history outperformed the remaining variants and also FPE baseline. This could be because of multiple reasons, and we discuss them below.

Firstly, contextual embedding, in place of static BPE embedding in FPE, of articles and comments is a key factor behind the superior performance. Specifically, the contextual embedding of article history with target article (reading track) and with corresponding comment history (writing track) has enabled us to generate a better representation of the input text.

Secondly, the dynamic FPE model, unlike the static FPE model, creates reading and writing fingerprints through GRU networks separately before concatenating them for sentiment prediction. With an extra GRU hidden layer in the reading track compared to the writing track, we have been able to create a higher level of understanding of the temporal relationship between target article and history articles. From these fingerprints, we also found that users with the closest fingerprints in the euclidean space are found to have higher prediction accuracy than that of farther fingerprint users.

Further, the ELECTRA-based model outperforms DistilBERT-based model in most of the cases, despite having only 20% of the number of parameters of DistilBERT-based model. This reiterates the result of Clark et al., 2020 that novel pretraining by a discriminative model that predicts whether each token in the corrupted input was replaced by a generator sample or not performs better than masked language modeling pretraining method of DistilBERT, even for a small model.

Other important inferences are:

1. Performance increases with length of relevant history for dynamic FPE models in general;
2. Articles with a large proportion of comments with negative sentiment have a higher micro F1-score compared to controversial (equally positive and negative sentiment comments) and positive comments (predominant positive sentiment comments).
3. BERT variant models with frozen parameters have better performance than those with all parameters trained. This may be because the pretrained model is trained on a much larger corpus than our problem dataset and so it has better language understanding.

For complete details of these experiments, please refer to longer version of the paper.

6 Limitations and future work

According to Sutrop and Laas-Mikko, 2012, fingerprinting for predicting behavior is second-generation biometrics, which is different from first-generation biometrics that uses characteristics that are visible to the naked eye such as facial images.
hand fingerprints. In this context, user fingerprinting in our model can be loosely classified under behavior fingerprinting. We did linguistic opinion and content-based user fingerprinting as a response history embedding for a user. In this section, we briefly discuss the limitations of the dataset and model, and future research direction.

Firstly, the newspaper articles, in general, may be biased in terms of story selection, tone, and organizing of the story. The users (readers) may also have an implicit bias - attitudes that unconsciously affect individual thoughts and actions - and confirmation bias - the tendency to support information that confirms their beliefs. To address these biases, we would like to extend our model to datasets that are not related to news articles.

Moreover, representations encoded in the models often inadvertently perpetuate undesirable social biases from the data on which they are trained. NLP models, especially neural embeddings, may perpetuate these biases towards race, religion, gender and disability (Hutchinson et al., 2020; Manzini et al., 2019; Sap et al., 2019). Though the BERT variant based sentence encoders exhibit less bias than previous models (May et al., 2019), we would also like to experiment with other sentence encoders to measure the bias in our predictions in future work.

Another limitation of our approach is that we used only article titles rather than whole content. This would be more critical when the title is misleading, for instance in satirical articles. Moreover, we have not experimented with multilingual models of pretrained BERT variants.

For future research, these experiments can be extended for whole article content and use various attention mechanisms to generate better fingerprints and also generate author profiles based on their reading history. Further, BERT variants trained in other languages can also be used.

7 Conclusion

In this paper, we propose a novel dynamic fingerprinting technique based on BERT variants and RNN networks to predict a user’s sentiment to an unseen article based on reading-writing history. Two variants of our model extract relevant history in two different ways and create contextual embedding for articles read by a user conditioned with target article and also corresponding comments. Finally, we used RNN to interpret temporal relationship and create a fingerprint, which is used to predict unseen target article. Our models demonstrated state-of-the-art performance on a real-world dataset. From our experiments, we found that performance saturates after an optimum length of relevant history.

Acknowledgments

Research was supported in part by grants NSF 1838147, NSF 1838145, ARO W911NF-20-1-0254. The views and conclusions contained in this document are those of the authors and not of the sponsors. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. Finally, we are greatly appreciative of the anonymous reviewers for their time and constructive comments.

References


Can Multilingual Transformers Fight the COVID-19 Infodemic?

Lasitha Uyangodage‡, Tharindu Ranasinghe§, Hansi Hettiarachchi♥, ‡University of Münster, §University of Wolverhampton, ♥Birmingham City University
luyangod@uni-muenster.de

Abstract
The massive spread of false information on social media has become a global risk especially in a global pandemic situation like COVID-19. False information detection has thus become a surging research topic in recent months. In recent years, supervised machine learning models have been used to automatically identify false information in social media. However, most of these machine learning models focus only on the language they were trained on. Given the fact that social media platforms are being used in different languages, managing machine learning models for each and every language separately would be chaotic. In this research, we experiment with multilingual models to identify false information in social media by using two recently released multilingual false information detection datasets. We show that multilingual models perform on par with the monolingual models and sometimes even better than the monolingual models to detect false information in social media making them more useful in real-world scenarios.

1 Introduction
By June 2021, the coronavirus (COVID-19) pandemic has affected 219 nations around the world with 176 million total cases and 3.81 million deaths. The nature of the virus caused many governments to implement lockdown in their countries. As a result, many people started spending more time at home during the pandemic and started using social media more, providing an unexpected boost to engagement on these platforms (Hettiarachchi and Ranasinghe, 2020b).

As a drawback of these exponential growths, social media has become a conduit for spreading both rumours and deliberate misinformation, and many perpetrators are deploying sites such as Facebook, Twitter, YouTube, and WhatsApp to create a sense of panic and confusion. On the other hand, the general public can not completely ignore the information seen in social media due to the fact that the Centers for Disease Control and Prevention, the World Health Organisation (WHO), numerous journals, government and other health care organisations are regularly posting guidance across a host of platforms. Therefore, rather than completely disregarding information seeing in social media, accurate identification of false information is crucial (Nguyen et al., 2020).

Considering the high data generation in social media, manual approaches to filter false information require significant human efforts. Therefore an automated technique to tackle this problem will be invaluable to the community. In the light of this many shared task has been organised to tackle the false information detection in social media (Shaar et al., 2021; Nakov et al., 2021a) leading to implement various machine learning models which can identify false information automatically (Uyangodage et al., 2021; Tziafas et al., 2021). However, most of these approaches build language-specific models trained specifically on a particular language. Given the fact that most of the social media platforms are massively multilingual, maintaining fake news identification models for each language would not be feasible. One machine learning model that can work across many languages would be invaluable to the community.

In this research, we explore multilingual models for false information detection. We experiment with two recently created datasets that target two different aspects in false information detection which also covers 5 languages; Arabic, Bulgarian, English, Spanish and Turkish. We show that multilingual models based on pretrained transformer models perform on par with the language-specific models trained for each language on both aspects in false information detection making them more feasible in real-world applications.
2 Related Work

False Information Detection Identifying false information in social media has been a major research topic in recent years. According to literature, mainly, there are two types of methods for false information detection as Social Context-based methods and Content-based methods (Guo et al., 2020). Social Context-based methods use different properties in user profiles such as user’s credibility (Li et al., 2019) or stances (Mohammad et al., 2017) while the Content-based methods use different features in the content of posts such as certain keywords, number of URLs and the length of textual content to detect false information. However, due to ethical considerations, most of the social media platforms do not allow to release datasets with details which can be used to identify users of the posts. Therefore Social Context-based methods have not been popular in recent research as Content-based methods. Due to the nature of datasets we use for this research, we also focused on Content-based methods.

Content-based methods mainly focus on different features of post contents. For example, Castillo et al. (2011) found that highly credible tweets have more URLs and lengthy textual contents than low credible tweets. Also, many studies utilise lexical and syntactic components of the content as useful features. For instance, Qazvinian et al. (2011) found part of speech (POS) as a distinguishable feature for false information detection. Similarly, Kwon et al. (2013) found that some types of sentiments including positive words (e.g. love, nice, sweet), negating words (e.g. no, not, never), cognitive action words (e.g. cause, know) and inferring action words (e.g. maybe, perhaps) as apparent features for a periodic time-series model to identify key linguistic differences between true and fake tweets. With the recent popularity gained by embedding and deep learning-based approaches in natural language processing, there was a tendency to use deep neural networks powered by content embeddings to perform false information classification too (Ma et al., 2016). Later, with the introduction of transformers (Devlin et al., 2019; Conneau et al., 2020), there was a tendency to involve large pretrained transformer models also (Uyangodage et al., 2021; Tziafas et al., 2021; Qarqaz et al., 2021). However, all of these models were trained specifically on a single language making them less useful in real scenarios where we need to process multilingual data.

Multilingual models Multilingual models allow training a single model to perform a task on multiple languages. These types of models have been used by many tasks such as offensive language identification (Ranasinghe and Zampieri, 2020, 2021a,b) and machine translation (Nguyen and Chiang, 2017; Aharoni et al., 2019). All of these studies train one machine learning model on all the languages which the training data is available and show that the multilingual models perform on par with or sometimes even better than monolingual models. The recently released multilingual transformer models that support more than 100 languages like BERT (Devlin et al., 2019), XLM-RoBERTa (Conneau et al., 2020) have made multilingual research easier. Even though these multilingual models improve the feasibility of the research to be applied on a real-world application, to the best of our knowledge, no prior work has been done for multilingual false information identification focused by this paper.

3 Data

For this research, we used two recently released datasets on false information identification. We mainly considered two factors when selecting datasets; the dataset should be annotated in multiple languages and it should have been annotated very recently.

The first dataset (NLP4IF) which was released for NLP4IF shared task; Fighting the COVID-19 Infodemic is about predicting several binary properties of a tweet on COVID-19 such as whether it is harmful, whether it contains a verifiable claim, whether it may be of interest to the general public and whether it appears to contain false information (Shaar et al., 2021). The data has been released for three languages; English, Arabic and Bulgarian1. Seven labels were targeted by this dataset. The first label was Verifiable Factual Claim: Does the tweet contain a verifiable factual claim?. We only considered this label for our research as this is directly related to false information detection and this label had the most annotated data out of the seven labels. False information detection using this label can be considered as a binary text classification task.

1The dataset can be downloaded from https://gitlab.com/NLP4IF/nlp4if-2021
The second dataset (CLEF2021) that we considered was released for CLEF2021 CheckThat-Lab (Nakov et al., 2021a) Task 1: Check-Worthiness Estimation of the tweets (Nakov et al., 2021b). Given a tweet, the participants need to predict whether it is worth fact-checking. This task is directly related to the first label of the NLP4IF dataset. However, contrast to the binary classification in the previous task, the models in this task need to predict a continuous value between 0-1 that reflects the worthiness to perform fact-checking. The dataset has been annotated in five languages; Arabic, Bulgarian, English, Spanish and Turkish (Nakov et al., 2021b) promoting multilingual research.

4 Architecture

The main motivation for our architecture is the recent success that the transformer models had in various natural language processing tasks including text classification (Ranasinghe and Zampieri, 2020, 2021a,b), word sense disambiguation (Hettiarachchi and Ranasinghe, 2020a, 2021), language identification (Jauhiainen et al., 2021) etc. Apart from providing strong results compared to RNN based architectures (Ranasinghe et al., 2019), transformer models like BERT (Devlin et al., 2019) provide pretrained multilingual language models that support more than 100 languages which will solve the multilingual issues of these tasks (Ranasinghe and Zampieri, 2020).

Transformer models take an input of a sequence and output the representations of the sequence. There can be one or two segments in a sequence which are separated by a special token [SEP] (Devlin et al., 2019). In this approach we considered a tweet as a sequence and no [SEP] token is used. Another special token [CLS] is used as the first token of the sequence which contains a special classification embedding. For text classification tasks, transformer models take the final hidden state of the [CLS] token as the representation of the whole sequence (Sun et al., 2019). A simple softmax classifier is added to the top of the transformer model to predict the probability of a class. For text regression tasks, a fully-connected layer is added on top of the [CLS] token. The fully-connected layer will have a single output neuron which predicts the target. For both tasks, all the parameters of the transformer model as well as W are fine-tuned jointly by maximising the log-probability of the gold truth.

5 Experimental Setup

We trained a transformer model for each dataset mentioned in Section 3. Given the very unbalanced nature of the datasets, the transformer models tend to overfit and predict only the majority class. Therefore, for each label, we took the number of instances in the training set for the minority class and undersampled the majority class to have the same number of instances as the minority class.

We then divided this undersampled dataset into a training set and a validation set using the 0.8:0.2 split. We mainly fine-tuned the learning rate and the number of epochs of the classification model manually to obtain the best results for the validation set. We obtained $1e^{-5}$ as the best value for the learning rate and 3 as the best value for the number of epochs for both datasets. The other configurations of the transformer model were set to a constant value over all the experiments in order to ensure consistency between them. We used a batch size of 8, Adam optimiser and a linear learning rate warm-up over 10% of the training data. The models were trained using only training data. We performed early stopping if the evaluation loss did not improve over 10 evaluation rounds. The implementation was done using HuggingFace transformer implementation (Wolf et al., 2020). A summary of hyperparameters and their values used to obtain the reported results are mentioned in Table 1. The optimised hyperparameters are marked with $\dagger$ and their optimal values are reported.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>learning rate$^1$</td>
<td>$1e^{-5}$</td>
</tr>
<tr>
<td>number of epochs$^1$</td>
<td>3</td>
</tr>
<tr>
<td>adam epsilon</td>
<td>$1e^{-8}$</td>
</tr>
<tr>
<td>warmup ration</td>
<td>0.1</td>
</tr>
<tr>
<td>warmup steps</td>
<td>0</td>
</tr>
<tr>
<td>max grad norm</td>
<td>1.0</td>
</tr>
<tr>
<td>max seq. length</td>
<td>120</td>
</tr>
<tr>
<td>gradient accumulation steps</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1: Hyperparameter specifications

For monolingual experiments, we trained language-specific transformer models on that particular language only. As pretrained transformer models, we used Arabert (Antoun et al., 2020) for Arabic, bert-base-cased (Devlin et al., 2019) for English, BETO: Spanish BERT for Spanish (Cañete et al., 2020) and BERTurk for Turkish. Unfortunately for Bulgarian, we could not find a suitable
pretrained transformer model. Therefore, for Bulgarian, we used the bert-multilingual-cased (Devlin et al., 2019) model.

For multilingual experiments, we first combined data instances from all the languages of each task which left us with two large multilingual false information identification datasets. Then we trained the transformer models on that combined datasets. As the multilingual pretrained transformer model, we used the bert-multilingual-cased (Devlin et al., 2019) model.

6 Results

In Table 2 we show the results we got for the test set of the NLP4IF dataset. We used the same evaluation metric as the organisers of the task; Macro F1 in order to compare our approach with the baselines and the best systems submitted.

<table>
<thead>
<tr>
<th>Language</th>
<th>Model</th>
<th>Macro F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>Monolingual</td>
<td>0.852</td>
</tr>
<tr>
<td></td>
<td>Qarqaz et al. (2021)</td>
<td>0.843</td>
</tr>
<tr>
<td></td>
<td>Multilingual</td>
<td>0.802</td>
</tr>
<tr>
<td></td>
<td>Random Baseline</td>
<td>0.552</td>
</tr>
<tr>
<td></td>
<td>Ngram Baseline</td>
<td>0.510</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>Multilingual</td>
<td>0.956</td>
</tr>
<tr>
<td></td>
<td>Shaar et al. (2021)</td>
<td>0.887</td>
</tr>
<tr>
<td></td>
<td>Monolingual</td>
<td>0.647</td>
</tr>
<tr>
<td></td>
<td>Random Baseline</td>
<td>0.594</td>
</tr>
<tr>
<td></td>
<td>Ngram Baseline</td>
<td>0.909</td>
</tr>
<tr>
<td></td>
<td>Ngram Baseline</td>
<td>0.842</td>
</tr>
<tr>
<td></td>
<td>Tziafas et al. (2021)</td>
<td>0.835</td>
</tr>
<tr>
<td></td>
<td>Monolingual</td>
<td>0.819</td>
</tr>
<tr>
<td></td>
<td>Ngram Baseline</td>
<td>0.647</td>
</tr>
<tr>
<td></td>
<td>Random Baseline</td>
<td>0.552</td>
</tr>
</tbody>
</table>

Table 2: Results ordered by Macro F1 for Arabic, Bulgarian and English languages in NLP4IF dataset. Monolingual implies the results of the monolingual models and Multilingual implies the results of the multilingual model for each language. Additionally, we report Ngram and Random baselines, and best systems submitted for the shared task.

As can be seen in the results, for the NLP4IF dataset, multilingual models perform better than the monolingual models and the best systems in Bulgarian and English while performing on par in Arabic. Please note that these best systems (Qarqaz et al., 2021; Tziafas et al., 2021) have been trained specifically on those language pairs using language specific natural language processing pipelines, yet the multilingual models outperform them in English and Bulgarian.

The results for CLEF2021 dataset is shown in Table 3. For this dataset also we used the same evaluation metric that the organisers used - Mean Average Precision (MAP) (Nakov et al., 2021b) ².

<table>
<thead>
<tr>
<th>Language</th>
<th>Model</th>
<th>MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>Best System</td>
<td>0.658</td>
</tr>
<tr>
<td></td>
<td>Monolingual</td>
<td>0.651</td>
</tr>
<tr>
<td></td>
<td>Multilingual</td>
<td>0.647</td>
</tr>
<tr>
<td></td>
<td>Ngram Baseline</td>
<td>0.428</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>Best System</td>
<td>0.737</td>
</tr>
<tr>
<td></td>
<td>Monolingual</td>
<td>0.711</td>
</tr>
<tr>
<td></td>
<td>Multilingual</td>
<td>0.700</td>
</tr>
<tr>
<td></td>
<td>Ngram Baseline</td>
<td>0.588</td>
</tr>
<tr>
<td>English</td>
<td>Best System</td>
<td>0.224</td>
</tr>
<tr>
<td></td>
<td>Monolingual</td>
<td>0.196</td>
</tr>
<tr>
<td></td>
<td>Multilingual</td>
<td>0.188</td>
</tr>
<tr>
<td></td>
<td>Ngram Baseline</td>
<td>0.052</td>
</tr>
<tr>
<td>Spanish</td>
<td>Best System</td>
<td>0.537</td>
</tr>
<tr>
<td></td>
<td>Monolingual</td>
<td>0.522</td>
</tr>
<tr>
<td></td>
<td>Multilingual</td>
<td>0.508</td>
</tr>
<tr>
<td></td>
<td>Ngram Baseline</td>
<td>0.450</td>
</tr>
<tr>
<td>Turkish</td>
<td>Best System</td>
<td>0.581</td>
</tr>
<tr>
<td></td>
<td>Monolingual</td>
<td>0.565</td>
</tr>
<tr>
<td></td>
<td>Multilingual</td>
<td>0.555</td>
</tr>
<tr>
<td></td>
<td>Ngram Baseline</td>
<td>0.354</td>
</tr>
</tbody>
</table>

Table 3: Results ordered by Mean Average Precision (MAP) for Arabic, Bulgarian, English, Spanish and Turkish languages in CLEF2021 dataset. Monolingual implies the results of the monolingual models and Multilingual implies the results of the multilingual model for each language. Best system denotes the results of the best system submitted to the language. Additionally, we report the Ngram baseline.

As can be seen in the results multilingual models outperformed monolingual models in Arabic, Bulgarian, Spanish and Turkish languages while performing on par with English. Similar to the results of the previous dataset, these multilingual models are very competitive with the best systems submitted to each of the languages.

7 Conclusion

In this paper, we explored multilingual models for false information identification using two recently created datasets. In our experiments, we observed that multilingual models built using powerful pretrained multilingual transformers perform on par or sometimes even better than the monolingual models. These results are consistent with

²The results are extracted from https://gitlab.com/checkthat_lab/clef2021-checkthat-lab
both datasets and across five languages. Findings in this paper would be valuable when building real-world applications for false information identification where maintaining separate machine learning models for each language would be more expensive and chaotic.

As future work, we would like to expand this research into more transformer models and more languages. We would like to experiment with how the multilingual transformer models with the cross-lingual concepts like XLM-RoBERTa would perform in multilingual false information identification. Furthermore, we would explore zero-shot and few-shot learning with multilingual models which would be beneficial to low resource language where the training data is scarce.

References


Contextual-Lexicon Approach for Abusive Language Detection

Francielle Vargas*†, Fabiana Góes*, Isabelle Carvalho‡
Fabricio Benevenuto†, Thiago A. S. Pardo*

*Institute of Mathematical and Computer Sciences, University of São Paulo, Brazil
‡Ribeirão Preto Medical School, University of São Paulo, Brazil
†Computer Science Department, Federal University of Minas Gerais, Brazil

{francielleavargas, fabianagoes, isabelle.carvalho}@usp.br
fabricio@dcc.ufmg.br, taspardo@icmc.usp.br

Abstract

Since a lexicon-based approach is more elegant scientifically, explaining the solution components and being easier to generalize to other applications, this paper provides a new approach for offensive language and hate speech detection on social media, which embodies a lexicon of implicit and explicit offensive and swearing expressions annotated with contextual information. Due to the severity of the social media abusive comments in Brazil, and the lack of research in Portuguese, Brazilian Portuguese is the language used to validate the models. Nevertheless, our method may be applied to any other language. The conducted experiments show the effectiveness of the proposed approach, outperforming the current baseline methods for the Portuguese language.

1 Introduction

In Brazil, hate speech is prohibited. Nevertheless, in government and civil society, the regulation of hate speech is not effective due to the difficulty to identify, quantify and classify abusive comments. Indeed, this is rather a difficult requirement to satisfy. According to Mesquita (2018), the Safer-net non-governmental organization, which operates in cooperation with public organizations in Brazil, as well as companies, such as Google, Facebook, and Twitter, proposed a collection of data on actions that violate human rights. The data is very worrisome: during the 2018 year’s election period, denunciations with xenophobia content had an increase of 2,369.5%; apology and public incitement to violence and crimes against life, 630.52%; neo-nazism, 548.4%; homophobia, 350.2%; racism, 218.2%; and religious intolerance, 145.13% ¹. Figure 1 shows the hate crimes evolution that occurred in the most populous Brazilian state ². The data was collected from São Paulo public security government. The pink line provides data on religious intolerance crimes, red on homophobia/transphobia, blue on race/ethnicity/color, green on region/origin, yellow on political intolerance, and light green on other crimes.

Figure 1: Hate crimes occurrence in São Paulo from 2016 to beginning of 2020.

Indeed, it is generally agreed that the high incidence of hate crimes is boosted by the popularization of online social networks. In social media, people and organizations may use the language to defamation, oppression, and terrorism. The language used intentionally in order to disrespect, insult or attack the reader is denominated in literature by abusive language, unless otherwise stated such as offensive language (Çöltekin, 2020; Pitenis et al., 2020; Razavi et al., 2010), hate speech (Schmidt and Wiegand, 2017; Waseem and Hovy, 2016) and cyberbullying (Rosa et al., 2019).

According to Warner and Hirschberg (2012), hate speech is a particular form of abusive lan-

¹https://www.bbc.com/portuguese/brasil-46146756

²https://www.ssp.sp.gov.br/
guage considering stereotypes to express an ideology of hate. In the same settings, Nockleby (2000) defines hate speech as “any communication that disparages a person or a group based on some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristic”.

To the best of our knowledge, no previous methods exist in order to embody an offensive lexicon annotated with contextual information to automatically classify abusive language on social media. Therefore, the main contribution of this paper is providing a new method for abusive comment detection on social media. Moreover, as already mentioned, due to the severity of the hate crimes in Brazil, and the lack of research in this language, Brazilian Portuguese is the language used to evaluate the proposed method, which showed high performance, outperforming the current baseline methods for Portuguese. Despite the proposed approach has been orchestrated over Brazilian Portuguese comments, the method in this paper may be applied to any other language. Finally, this paper also presents the evaluation of algorithms used for feature selection.

The remainder of the paper is structured as follows. In Section 2, we briefly introduce the most relevant related work. Section 3 presents a overview of the data. Sections 4 and 5 describe the proposed method and the performed experiments. In Section 6, we report the evaluation results. In Section 7, we make some final remarks.

2 Related Work

Several efforts have been made to provide automated detection approaches for hate speech and offensive languages on social media (Gao and Huang, 2017; Davidson et al., 2017; Warner and Hirschberg, 2012). The basic state of the art framework consists of creating lists of words that contain sets of known hate keywords. Furthermore, corpora are manually annotated in order to construct training datasets labeled with hate speech and non-hate speech. At last, automated methods of learning, such as traditional machine learning or neural-based machine learning, are used to automatically detect hate speech in social media texts. However, most hate speech resources and models are proposed for English (Zampieri et al., 2019; Basile et al., 2019; Davidson et al., 2017; Njagi et al., 2015; Ting et al., 2013).

For Portuguese, Fortuna et al. (2019) adopted the definition of hate speech proposed by Fortuna and Nunes (2018), and proposed a new dataset composed of 5,668 tweets, as well as automated methods using a hierarchy of hate to identify social groups of discrimination. The authors have obtained 78% f1-score using a neural network (LSTM). Additionally, de Pelle and Moreira (2017) provide a new dataset composed of 1,250 comments collected from G1 Brazilian online newspaper and annotated with offensive and non-offensive tags. In addition, the authors present classification results achieved by classical machine learning algorithms (SMV and NB), reporting results over 81% f1-score.

3 Data Overview

3.1 HateBR Corpus

HateBR was proposed by Vargas et al. (2021), and consists of the first large-scale dataset for hate speech and offensive language detection for the Portuguese language. HateBR corpus annotation presents 89% of human inter-annotator agreement. The corpus is composed of 7,000 Instagram comments annotated with three different layers: (i) binary classes (offensive and non-offensive); (ii) offense-levels (highly, moderately, and slightly offensive); and (iii) nine hate group targets (xenophobia, racism, homophobia, sexism, religious intolerance, partyism, apology to dictatorship, antisemitism, and fatphobia).

The authors report that the comments were collected from six public Instagram accounts of the Brazilian political domain. Moreover, they selected three liberal-party accounts followed by three conservative-party accounts, being four women and two men. Due to the degree of complexity of the offensive language and hate speech detection tasks, mainly because it involves a highly politicized domain, the authors decided to enroll annotators at higher levels of education (Ph.D.), which are from different political orientations and colors in order to minimize bias.

Tables 1, 2, 3 show the HateBR dataset statistics.

1 According to the professor at Harvard University, “partyism” is a form of hostility and prejudice that operates across political lines (Sunstein, 2016). Moreover, Westwood et al. (2018) demonstrated that partyism influences behaviors and non-political judgments.
Table 1: Binary class: offensive x non-offensive.

<table>
<thead>
<tr>
<th>Binary Class</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Offensive</td>
<td>3,500</td>
</tr>
<tr>
<td>Offensive</td>
<td>3,500</td>
</tr>
<tr>
<td>Total</td>
<td>7,000</td>
</tr>
</tbody>
</table>

Table 2: Offense levels.

<table>
<thead>
<tr>
<th>Offense-levels Classes</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slightly Offensive</td>
<td>1,281</td>
</tr>
<tr>
<td>Moderately Offensive</td>
<td>1,440</td>
</tr>
<tr>
<td>Highly Offensive</td>
<td>779</td>
</tr>
<tr>
<td>Total</td>
<td>3,500</td>
</tr>
</tbody>
</table>

Table 3: Hate group targets.

<table>
<thead>
<tr>
<th>Hate Groups</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partyism</td>
<td>496</td>
</tr>
<tr>
<td>Sexism</td>
<td>97</td>
</tr>
<tr>
<td>Religious Intolerance</td>
<td>47</td>
</tr>
<tr>
<td>Apology to Dictatorship</td>
<td>32</td>
</tr>
<tr>
<td>Fat Phobia</td>
<td>27</td>
</tr>
<tr>
<td>Homophobia</td>
<td>17</td>
</tr>
<tr>
<td>Racism</td>
<td>8</td>
</tr>
<tr>
<td>Antisemitism</td>
<td>2</td>
</tr>
<tr>
<td>Xenophobia</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>727</td>
</tr>
</tbody>
</table>

3.2 MOL - Multilingual Offensive Lexicon

MOL (Multilingual Offensive Lexicon)\(^4\) consists of a multilingual offensive lexicon, composed of 1,000 explicit and implicit offensive and swearing expressions of offense and swearing, which were annotated with a binary class: context-dependent and context-independent offensive. For example, the term “*vadia*” (“slut”) consists of a context-independent offensive term. On other hand, the term “*inútil*” (“useless”) is a context-dependent offensive term. Note that this last term is classified as context-dependent offensive because it also may be employed in a non-offensive context, such as “this smartphone is useless” or “the process is useless for this task”.

The MOL was extracted from HateBR corpus (Vargas et al., 2021), and each term or expression was annotated by three different annotators obtaining a high human agreement score (73% Kappa). Furthermore, as already mentioned, implicit content also was extracted using “clue terms or expressions”. For example, the expression “*voltar para a jaula*” (“go back to the cage”) consists of a “clue expression” to identify the implicit offensive term “*ladrao*” (“thief”). Finally, terms that showed explicit potential to indicate hate speech targets were also annotated, for instance, “*vadia*” (“slut”) and “*judeus dos infernos*” (“jews from hell”). Note that the occurrence of these cases may indicate sexist and antisemitism comments.

4 The Proposed Approach

We present a new approach to detect abusive comments on the web and social media. Our method embodies an offensive lexicon, which provides contextual information on hate speech and offenses. We show in detail our approach in Sections 4.1, 4.2, and 4.3.

4.1 Tasks

In this paper, we assume that abusive language detection may be divided into two main tasks: (i) offensive language detection, (ii) hate speech detection. Considering this premise, we train two different classifiers. The first classifier automatically identifies offensive comments. On the other hand, the second classifier automatically identifies comments that present hate speech content. Note that a hate speech comment is always an offensive comment, however, an offensive comment may present or not hate speech content. Figure 2 shows each of these different tasks in detail.

4.2 The Feature Set

Defining the most appropriate textual representation is a crucial task that directly influences the performance of the predictive model built by the classification algorithms. In this paper, we modeled hate speech and offensive language through

![Figure 2: Our approach to use the HateBR dataset and automatically detect offensive comments, as well as offensive comments with hate speech content.](https://github.com/francielleavargas/MOL)
different representation paradigms and features. We describe each one in what follows.

4.2.1 Lexical and Morphosyntactic Features
We selected lexical elements (each word into the document without stopwords), as well as part-of-speech-based features, using the Stanford Stanza POS tagger\(^5\) for Portuguese.

4.2.2 Lexicon-Based Features
We included features from three different lexicons: one sentiment lexicon (Sentilex-PT (Silva et al., 2012)), one emotion lexicon (WordNetAffect.BR (Pasqualotti, 2008)), and finally, one offensive contextual lexicon (MOL).

1. Sentiment Lexicon: we evaluated features based on sentiment (Silva et al., 2012) and emotion (Pasqualotti, 2008) lexicons, which present semantic polarity (e.g., positive, negative and neutral) and emotion types (e.g., anger, love, hate, disgust, suspicious and fear).

2. Contextual Lexicon: we evaluated features based on an offensive lexicon (MOL) annotated with contextual (context-dependent and context-independent) labels.

4.2.3 Word Embedding Features
We also evaluated word embedding-based features. Different from other language models, BERT (Bidirectional Encoder Representations from Transformers) is usually used to pre-train deep bidirectional representations from unlabeled texts by jointly conditioning on both left and right contexts in all internal network layers (Devlin et al., 2019). In a similar setting, we also used fastText, the Facebook pre-trained models (Joulin et al., 2016).

4.2.4 Feature Set Overview
We summarize in Table 4 the five feature representations used in this paper.

<table>
<thead>
<tr>
<th>N.</th>
<th>Features</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>POS+S</td>
<td>Bag-of-POS+Sentiment</td>
</tr>
<tr>
<td>2</td>
<td>BOW</td>
<td>Bag-Of-Words</td>
</tr>
<tr>
<td>3</td>
<td>MOL</td>
<td>Bag-Of-MOL</td>
</tr>
<tr>
<td>4</td>
<td>B+M</td>
<td>Bag-Of-Words embodying the MOL</td>
</tr>
<tr>
<td>5</td>
<td>BERT &amp; fastText</td>
<td>Multilingual pre-trained models</td>
</tr>
</tbody>
</table>

2. BOW: we created a bag-of-words representation or, in other words, we generated a text representation that describes the occurrence of dataset vocabulary for each comment. We simply calculate how many times each word of our dataset vocabulary (features) appears in each comment.

3. MOL: in this representation, a bag-of-words was generated using the terms or expressions extracted from the offensive lexicon (MOL), which were used as features. Therefore, for each comment, the occurrence of the MOL’s terms was counted. Additionally, context labels (context-independent and context-dependent) have been considered in order to compute different weights to context-independent and context-dependent features. The frequency of the terms with context-independent labels were multiplied by 2, while the frequency of the terms with context-dependent labels remained the same. Specifically for hate speech detection task, we also checked if a term presented any markers that identify hate speech content, and, if this condition was true, an additional weight was accounted. Therefore, in the MOL representation, the value of a term \(x\) in the document (comment) \(y\) for the offensive comment detection (task 1) is defined according to

\[
MOL_{x,y} = freq_{x,y} \ast weightC_x
\]  

and for the hate speech detection (task 2) is given by

\[
MOL_{x,y} = freq_{x,y} \ast weightC_x \ast weightH_x
\]

where \(freq\) is the frequency of the term in the document, \(weightH = 2\) when the term is a marker that identifies hate speech and \(weightH = 1\) otherwise, \(weightC = 1\) for

\(^5\)https://stanfordnlp.github.io/stanza/pos.html
context-dependent terms and $weight_C = 2$
when the term is context-independent.

4. B+M: we generated a bag-of-words representation, which embodies context label information from the offensive lexicon (MOL). In other words, we firstly generated a bag-of-words from all comments into the dataset. Then, we performed the match with terms into MOL, and then we assigned a weight for terms or expressions labeled with context-dependent (weaker weight) and context-independent (stronger weight). The contextual labels are provided by MOL. Therefore, in B+M representation, the value of a term $x$ into the document (comment) $y$ is defined according to

$$B + M_{x,y} = \text{freq}_{x,y} \times weight_C$$  \hspace{1cm} (3)

where $freq$ is the frequency of the term in the document, $weight_C = 2$ for context-dependent terms and $weight_C = 3$ when the term is context-independent.

5. In a different setting, the feature extraction for the BERT and fastText followed state of the art text classification with a maximum size of 500. For the fastText classifier, we set the maximum size equal to 64 and the maximum number of features equal to 10,000. We used the standard processor model and evaluated the n-gram range for unigram, bigram, and trigram.

4.3 The Learning Methods

In general, previous works on hate speech detection use neural networks or traditional machine learning techniques on specific communities (Davidson et al., 2017; Founta et al., 2019; Del Vigna et al., 2017; Njagi et al., 2015; Djuric et al., 2015). In order to evaluate the performance of neural networks and traditional machine learning techniques, we used the following learning methods: Support Vector Machine (SVM) (Scholkopf and Smola, 2001) with linear kernel; Multinomial Naive Bayes (NB) (McCallum et al., 1998; Eyheramendy et al., 2003); Multilayer Perceptron (MLP) (Haykin, 2009) with one hidden layer (with 100 neurons), and ReLU activation function; Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) with two hidden layers (with 200 and 50 neurons, respectively) and a softmax output unit for the binary classification. ReLU was used as the activation function, as well as number of epochs equal to 10, and a random batch size of 100 documents. Moreover, we also used pre-trained models of word embeddings, such as BERT (Devlin et al., 2019) and fastText (Joulin et al., 2016).

5 Experiments

We carried out a wide variety of experiments. We describe the entire process in Sections 5.1, 5.2, 5.3 and 5.4.

5.1 Data Preparing

We accomplished an approach for data preparation, as shown in Figure 3.

![Figure 3: Data preparation.](https://spacy.io/models/pt)

Firstly, we normalized our dataset using the normalization tool for Brazilian Portuguese proposed by Bertaglia and Nunes (2016). The normalization process consists of identifying noise, which is very common in User-Generated Content (UGC), such as orthographic errors, often phonetically-motivated, abbreviations and expressions often used informally by web users, proper names and acronyms wrongly or not at all capitalized, agglutinated words that should be split, and wrong use of sentence delimiters; and suggesting possible substitutions.

Moving forward, in the second step, we remove emoticons, special characters, accounts, hyperlinks, and websites. In step 3, we lemmatize our dataset using Spacy$^6$. Finally, in step 4, accentuation is removed.

5.2 Feature Selection

Feature selection (FS) allows the removal of irrelevant and redundant features. In this paper, in order to select the best feature set, we applied the following FS algorithms: (i) Correlation-based Feature Selection (CFS) (Hall, 1998), which selects characteristics that are highly correlated with

$^6$https://spacy.io/models/pt
the class and not correlated with each other using Pearson coefficient\(^7\) as criteria, and (ii) Information Gain Analysis (InfoGain) (Witten et al., 2016), which quantifies and chooses the characteristics that have the maximum information gain concerning the class. We apply both FS techniques on the NB, SVM, MLP and LSTM models. For BERT and fastText features, we do not apply FS techniques. Finally, we evaluated the performance of the FS techniques for each feature representation. More specifically, we measure the potential of the algorithms to help in the gain and loss of accuracy, precision, recall, and f1-score. Results are shown in Table 7.

5.3 Class Balancing

The most common class balancing methods are oversampling (Chawla et al., 2002) and undersampling (Witten et al., 2016). In undersampling, the number of examples of each class is maintained based on the number of examples from the minority class. Differently, in oversampling, the approach involves the construction of examples for the minority class, although these examples may not add any new information to the model. In our experiments, we adopted the undersampling on the unbalanced classes of hate speech, specially due to the fact that this approach makes overfitting unlikely. Note that, in our dataset (the HateBR), there are 727 labeled hate speech samples versus 2,227 labeled non-hate speech samples. As a result of the undersampling approach, we obtained 727 labeled samples for hate speech and 727 samples for non-hate speech.

5.4 Evaluation

Our models were trained and tested using 10-fold cross-validation (Stone, 1974). We have computed the classical machine learning evaluation measures of Precision, Recall and F1-Score. We present these evaluative measures for each class involved, as well as simple arithmetic means. The results are shown in Table 5. Moreover, we evaluated BERT and fastText pre-trained models, and show the obtained results in Table 6.

We also present the evaluation of the methods with feature selection (FS). We measure the gain and loss of precision, recall, and f1-score for each selected algorithm (CFS and InfoGain) in both tasks: offensive language detection and hate speech detection, as well as for each representations: POS+S, BOW, MOL and B+M. Table 7 shows the results. We should point out that T1 is the sum of each representation, and T2 is the sum for each FS algorithm.

Finally, Table 8 shows the comparison of the results with the current baseline methods for Portuguese.

6 Results

As shown in Table 5, the B+M proposed method in this paper obtained better results of precision, recall, and f1-score in both tasks - offensive language and hate speech detection. The worse results were obtained using the POS+S approach, which combines part-of-speech and sentiment lexicon features. We should point out the considerable impact of an offensive lexicon for abusive language detection, when compared to the impact of a sentiment lexicon. Our results showed that sentiment lexicon approach present weak performance for abusive language detection on the web and social media.

Moving forward, the conducted experiments also show that the traditional machine learning techniques presented better performance than neural-based classifiers for offensive language tasks. Nevertheless, for the hate speech detection task, the neural-based classifier overcame the traditional machine learning methods.

In general, BERT and fastText, as shown in Table 6, presented a high performance for both tasks (offensive language and hate speech detection), even though our approach (B+M) has overcome the fastText (trigrams) in 2% (f1-score) for hate speech detection, as well as presented better precision performance, and the same recall and f1-score performances for offensive language detection.

Considering the feature selection (FS) performance, as shown in Table 7, the InfoGain algorithm produced better results for precision, recall, and f1-score than CFS algorithm for offensive language detection (task 1). On other hand, for hate speech detection (task 2), CFS algorithm obtained better performance than InfoGain in recall and f1-score. Moreover, for offensive language detection (task 1), InfoGain applied on BOW and B+M representations obtained performance gain, and POS+S and MOL presented loss of perfor-

---

\(^7\)Pearson’s correlation coefficient is a linear correlation coefficient that returns a value between -1 and +1. A -1 means there is a strong negative relationship, and +1 means there is a strong positive relationship.
Table 5: NB, SVM, MLP and LSTM Evaluation.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Features set</th>
<th>Class</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>SVM</td>
<td>MLP</td>
<td>LSTM</td>
<td>NB</td>
<td>SVM</td>
</tr>
<tr>
<td>POS+S</td>
<td>0</td>
<td>0.50</td>
<td>0.51</td>
<td>0.47</td>
<td>0.49</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.50</td>
<td>0.51</td>
<td>0.54</td>
<td>0.49</td>
</tr>
<tr>
<td>Avg</td>
<td>0</td>
<td>0.50</td>
<td>0.51</td>
<td>0.51</td>
<td>0.49</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.85</td>
<td>0.82</td>
<td>0.72</td>
<td>0.83</td>
</tr>
<tr>
<td></td>
<td>Avg</td>
<td>0.85</td>
<td>0.88</td>
<td>0.86</td>
<td>0.85</td>
</tr>
<tr>
<td>BOW</td>
<td>0</td>
<td>0.74</td>
<td>0.78</td>
<td>0.94</td>
<td>0.79</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.95</td>
<td>0.94</td>
<td>0.72</td>
<td>0.93</td>
</tr>
<tr>
<td>Avg</td>
<td>0.85</td>
<td>0.86</td>
<td>0.83</td>
<td>0.86</td>
<td>0.81</td>
</tr>
<tr>
<td>MOL</td>
<td>0</td>
<td>0.84</td>
<td>0.84</td>
<td>0.91</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.93</td>
<td>0.93</td>
<td>0.81</td>
<td>0.85</td>
</tr>
<tr>
<td>Avg</td>
<td>0.89</td>
<td>0.88</td>
<td>0.86</td>
<td>0.85</td>
<td>0.88</td>
</tr>
<tr>
<td>B+M</td>
<td>0</td>
<td>0.52</td>
<td>0.49</td>
<td>0.42</td>
<td>0.52</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.52</td>
<td>0.47</td>
<td>0.63</td>
<td>0.52</td>
</tr>
<tr>
<td>Avg</td>
<td>0.52</td>
<td>0.48</td>
<td>0.53</td>
<td>0.52</td>
<td>0.52</td>
</tr>
<tr>
<td>POS+S</td>
<td>0</td>
<td>0.62</td>
<td>0.84</td>
<td>0.43</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.73</td>
<td>0.61</td>
<td>0.91</td>
<td>0.61</td>
</tr>
<tr>
<td>Avg</td>
<td>0.68</td>
<td>0.72</td>
<td>0.67</td>
<td>0.73</td>
<td>0.66</td>
</tr>
<tr>
<td>BOW</td>
<td>0</td>
<td>0.61</td>
<td>0.62</td>
<td>0.58</td>
<td>0.60</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.67</td>
<td>0.71</td>
<td>0.73</td>
<td>0.84</td>
</tr>
<tr>
<td>Avg</td>
<td>0.64</td>
<td>0.66</td>
<td>0.66</td>
<td>0.72</td>
<td>0.64</td>
</tr>
<tr>
<td>MOL</td>
<td>0</td>
<td>0.79</td>
<td>0.77</td>
<td>0.93</td>
<td>0.71</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.78</td>
<td>0.92</td>
<td>0.76</td>
<td>0.85</td>
</tr>
<tr>
<td>Avg</td>
<td>0.78</td>
<td>0.84</td>
<td>0.85</td>
<td>0.78</td>
<td>0.78</td>
</tr>
</tbody>
</table>

Table 6: BERT and fastText Evaluation.

<table>
<thead>
<tr>
<th>Models</th>
<th>Class</th>
<th>Task 1: Offensive Language Detection</th>
<th>Task 2: Hate Speech Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>F1-Score</td>
</tr>
<tr>
<td>BERT</td>
<td>0</td>
<td>0.85</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>Avg</td>
<td>0.86</td>
<td>0.86</td>
</tr>
<tr>
<td>fastText (unigram)</td>
<td>0</td>
<td>0.88</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>Avg</td>
<td>0.87</td>
<td>0.87</td>
</tr>
<tr>
<td>fastText (bigrams)</td>
<td>0</td>
<td>0.83</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>Avg</td>
<td>0.85</td>
<td>0.85</td>
</tr>
<tr>
<td>fastText (trigrams)</td>
<td>0</td>
<td>0.83</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td>Avg</td>
<td>0.86</td>
<td>0.86</td>
</tr>
</tbody>
</table>

mance. For hate speech detection (task 2), InfoGain applied on B+M representation presented performance gain. Differently from this, POS+S, BOW, and MOL had a loss of performance using InfoGain. Differently, CFS algorithm applied to BOW and B+M obtained performance gain, and when applied to POS+S and MOL representations, presented loss of performance.
Table 7: Feature selection performance.

<table>
<thead>
<tr>
<th>Measures</th>
<th>FS</th>
<th>Features set</th>
<th>Task 1: Offensive Language Detection</th>
<th>Task 2: Hate Speech Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Learning Methods</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NB</td>
<td>SVM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>T1</td>
<td>T2</td>
</tr>
<tr>
<td>Precision</td>
<td>CFS</td>
<td>POS+S</td>
<td>-0.25</td>
<td>-0.01</td>
</tr>
<tr>
<td></td>
<td>Info Gain</td>
<td>POS+S</td>
<td>-0.33</td>
<td>-0.01</td>
</tr>
<tr>
<td></td>
<td>CFS</td>
<td>ROW</td>
<td>-0.04</td>
<td>-0.05</td>
</tr>
<tr>
<td></td>
<td>Info Gain</td>
<td>ROW</td>
<td>-0.02</td>
<td>-0.03</td>
</tr>
<tr>
<td></td>
<td>CFS</td>
<td>MOL</td>
<td>-0.06</td>
<td>-0.04</td>
</tr>
<tr>
<td></td>
<td>Info Gain</td>
<td>MOL</td>
<td>-0.11</td>
<td>-0.06</td>
</tr>
<tr>
<td></td>
<td>CFS</td>
<td>B+M</td>
<td>-0.02</td>
<td>-0.03</td>
</tr>
<tr>
<td></td>
<td>Info Gain</td>
<td>B+M</td>
<td>-0.01</td>
<td>-0.05</td>
</tr>
<tr>
<td></td>
<td>CFS</td>
<td>BOW</td>
<td>-0.12</td>
<td>-0.05</td>
</tr>
<tr>
<td></td>
<td>Info Gain</td>
<td>BOW</td>
<td>-0.05</td>
<td>-0.05</td>
</tr>
<tr>
<td></td>
<td>CFS</td>
<td>POS+S</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Info Gain</td>
<td>POS+S</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>CFS</td>
<td>ROW</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Info Gain</td>
<td>ROW</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>CFS</td>
<td>MOL</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Info Gain</td>
<td>MOL</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>CFS</td>
<td>B+M</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Info Gain</td>
<td>B+M</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>CFS</td>
<td>BOW</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Info Gain</td>
<td>BOW</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### 6.1 Comparing Results

Table 8 shows a comparison of results between our new proposed method and baseline methods for Portuguese. Although a direct comparison is unfair (as the authors use different datasets), it offers an idea of the general performance of the methods.

de Pelle and Moreira (2017) report a f1-score of 81% using SVM and NB algorithms. For the same algorithms, our approach presented 88% of f1-score, improving the performance. In the same settings, Fortuna et al. (2019) report a f1-score of 78% using the LSTM algorithm. In our experiments, we obtained an f1-score of 86%, also using the LSTM algorithm, consequently, our approach presented better performance.

Table 8: Comparison of results.

<table>
<thead>
<tr>
<th>Dataset language</th>
<th>Algorithms</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our approach</td>
<td>Brazilian Portuguese SVM and NB</td>
<td>88%</td>
</tr>
<tr>
<td>de Pelle and Moreira (2017)</td>
<td>Brazilian Portuguese SVM and NB</td>
<td>81%</td>
</tr>
<tr>
<td>Our approach</td>
<td>Brazilian Portuguese LSTM</td>
<td>86%</td>
</tr>
<tr>
<td>Fortuna et al. (2019)</td>
<td>European and Portuguese LSTM</td>
<td>78%</td>
</tr>
</tbody>
</table>

### 7 Conclusions

In this work, we provide a new approach for the automatic detection of abusive comments on social media. Our approach embodies an offensive lexicon that provides contextual information. Due to the increase of abusive comments on social media in Brazil, as well as the lack of research in annotated datasets, we decided to use an Brazilian annotated dataset to evaluate the models. The proposed approach obtains high performances: 88% f1-score for offensive comments detection, and 85% for comments with hate speech, which overcame the current baseline methods for Portuguese.

We also evaluated the performance of feature selection (FS) methods, and conclude that InfoGain algorithm is the best algorithm for the offensive comment detection task, considering the obtained gains in recall and f1-score. For the hate speech detection task, CFS algorithm obtained better performance. Accordingly, based on the obtained results, we concluded that the proposed approach in this paper for automated detection of abusive comments is efficient and highly relevant, bearing in mind the current Brazilian social scenario, in which hateful comments are a very relevant social problem. Moreover, in the next year (2022), there will be presidential elections in Brazil, and this paper may provide a reliable automated approach for abusive comments detection in order to minimize political polarization, as well as hate crimes on social media.

### Acknowledgements

The authors are grateful to DCC-UFMG research project: WHATSAPP MONITOR - SIMP 18048 for supporting this work.
References


Comparative Analysis of Fine-tuned Deep Learning Language Models for ICD-10 classification task for Bulgarian Language

Boris Velichkov\textsuperscript{1,2}, Sylvia Vassileva\textsuperscript{1}, Simeon Gerginov\textsuperscript{1}, Boris Kraychev\textsuperscript{2}, Iyaylo Ivanov\textsuperscript{1}, Philip Ivanov\textsuperscript{1}, Ivan Koychev\textsuperscript{1,2} and Svetla Boytcheva\textsuperscript{3}

\textsuperscript{1} Faculty of Mathematics and Informatics, Sofia University "St. Kliment Ohridski", Sofia, Bulgaria
\texttt{sylvia.vassileva@gmail.com, simeonergginov1@gmail.com, sashovi@uni-sofia.bg, fsivanov@uni-sofia.bg, koychev@fmi.uni-sofia.bg}
\textsuperscript{2} GATE Institute, Sofia University "St. Kliment Ohridski", Sofia, Bulgaria
\texttt{boris.velichkov@gate-ai.eu, boris.kraychev@gate-ai.eu}
\textsuperscript{3} Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Sofia, Bulgaria
\texttt{svetla.boytcheva@iict.bas.bg}

Abstract

The task of automatic diagnosis encoding into standard medical classifications and ontologies is of great importance in medicine - both to support the daily tasks of physicians in the preparation and reporting of clinical documentation, and for automatic processing of clinical reports. In this paper, we investigate the application and performance of different deep learning transformers for automatic encoding in ICD-10 of clinical texts in Bulgarian. The comparative analysis attempts to find which approach is more efficient to be used for fine-tuning of pre-trained BERT family transformer to deal with a specific domain terminology on a rare language such as Bulgarian. On the one hand, we use SlavicBERT and MultiligualBERT models, which are pre-trained for a common vocabulary in Bulgarian but lack medical terminology. On the other hand, we compare them to BioBERT, ClinicalBERT, SapBERT, BlueBERT models, which are pre-trained for medical terminology in English, but lack training for language models in Bulgarian, and vocabulary in Cyrillic. In our research study, all BERT models are fine-tuned with additional medical texts in Bulgarian and then applied to the classification task for encoding medical diagnoses in Bulgarian into ICD-10 codes. A big corpus of diagnoses in Bulgarian annotated with ICD-10 codes is used for the classification task. Such an analysis gives a good idea of which of the models would be suitable for tasks of a similar type and domain. The experiments and evaluation results show that both approaches have comparable accuracy.

1 Introduction

The task for automatic encoding of Electronic Health Records (EHR) with standard medical classifications is a hot-topic. The international classification of diseases, 10th revision (ICD-10)\textsuperscript{1} is one of the most commonly used standard medical classifications due to the availability of translations in several languages. It is a hierarchical classification that encodes each diagnosis into a standard code which is used for statistical analysis and insurance reimbursement. The current solutions for this task are based on a restricted subset of ICD-10 codes or address some specific task trained on a small manually annotated corpus.

Recently some deep learning models like BERT (Devlin et al., 2018) pre-trained transformers were applied for different Natural Language Processing (NLP) and clinical NLP tasks. The first type of transformers is language models that cover common vocabulary on several languages: SlavicBERT (Arkhipov et al., 2019) and MultiligualBERT (Pires et al., 2019). The second type is transformers that cover specific terminology in English. In this particular case, the base language model is pre-trained with medical terminology by using scientific articles abstracts from PubMed\textsuperscript{2} or full-text articles from PMC\textsuperscript{3}: BioBERT (Lee et al., 2020), ClinicalBERT (Alsentzer et al., 2019), SapBERT (Liu et al., 2021), BlueBERT (Peng et al., 2019), MT-BERT (Peng et al., 2020), PubMedBERT (Gu et al., 2020). The later transformers prove that rel-

\textsuperscript{1}https://icd.who.int/browse10/2019/en
\textsuperscript{2}https://pubmed.ncbi.nlm.nih.gov/
\textsuperscript{3}https://www.ncbi.nlm.nih.gov/pmc/
atively high accuracy can be achieved in training for automatic ICD-10 classification task for the English language (Moons et al., 2020). We hypothesise that comparable accuracy can be achieved also for languages other than English using either type of pre-trained BERT transformers.

2 Related Work

The task for automatic ICD-10 encoding of textual descriptions of diagnosis was addressed in several research challenges like i2b2 NLP Challenges, CLEF eHealth, etc. The major problem is that ICD-10 classification contains more than 11K codes and requires a significant number of labeled training data. In general, there are only available labeled datasets for a limited number of ICD-10 codes, which is one of the reasons why this task is not yet solved to the full range of ICD-10 codes. Lavergne et al presented in (Lavergne et al., 2016) a dataset for for ICD-10 coding of death certificates that contains 377,677 labeled statements with 3,457 unique ICD-10 codes. Usually, the labeled datasets are highly unbalanced that has a huge impact on the annotation method performance. This problem was addressed in (Parlak and Uysal, 2018), where the authors apply techniques for imbalance effects reduction, like splitting feature spaces and compressing label dimension. The ICD-10 classification task was investigated for several languages. The best performance for languages other than English was achieved with SVM models (Bagheri et al., 2020) F1 54.9% for Dutch; the longest common subsequence problem (Chen et al., 2017) for Chinese with F1-score of 81.1%; a hierarchical approach (Ning et al., 2016) for Chinese with F1 score of 91.08%; information retrieval techniques for Turkish (CEYLAN et al., 2012) with the best score of 76.5%. Another approach is to view the problem as a multi-label classification task and use neural networks like CNN, LSTM/BiLSTM, and HA-GRU (Wang et al., 2020), or applying BERT which has shown good results on this task in German (Amin et al., 2019). Hybrid approaches, that combine different models show a slight improvement in the results (Amin et al., 2019).

For the Bulgarian language was done some preliminary experiments using SVM and small training corpora (Boytcheva, 2011), where the model achieved F-score 84%. We need to mention that the reported results in this work are based on significantly smaller training and test datasets with limited number of ICD-10 classes. In this paper we use big annotated corpora and include almost all ICD-10 codes used by medical practitioners in Bulgaria. In (Velichkov et al., 2020) we show some successful application of the BERT pretrained transformers for ICD-10 encoding. Inspired by the promising results we will investigate both language models: MultilingualBERT and SlavicBERT and will compare them with the state-of-the-art models for medical domain in English: BioBERT, ClinicalBERT, BlueBERT and SapBERT.

3 Data

3.1 Language Model Pre-training Dataset

For the Pre-training Dataset we have used a combination of medical articles and medical journals scraped from the internet. Medical articles were crawled from MedInfo4. We’ve also used a dataset of crawled medical articles that is already publicly available in GitHub5. Medical journals were crawled from MedUnion6, JournalsMuVarna7, MedicinaNauka8, CmlMuSofia9, Bulsem10, Basa11, MedSport12 and Vma13. The crawled medical articles and journals were cleaned from single and double quotes, as well as any special characters and new lines.

From MedInfo we have crawled 1,740 medical articles. Each article describes different topics in terms of medical diseases, as well as possible treatments for the different diseases.

Each medical journal is in a PDF format and was split by page during crawling.

From MedUnion we have crawled 612 pages of medical journals. Each journal describes modern medicine and different aspects of it.

From JournalsMuVarna we have crawled 1,230 pages of medical journals. Each journal describes different topics like Social medicine, Health policy, Healthcare management, History of medicine and healthcare, and others.

---

4 https://www.medinfo.bg/
5 https://github.com/BorisVelichkov/scraping-framar-and-bgmedic
6 http://www.medunion-bg.org/
7 https://journals.mu-varna.bg/index.php/sm/index
8 https://medicina.nauka.bg/
9 http://cml.mu-sofia.bg/CML/mpreg/index.html
10 http://www.bulsem.bg/bg/about-jem
11 https://www.basa.bg/
12 https://www.med-sport.net/index.html
13 https://www.vma.bg/
From MedicinaNauka we have crawled 281 pages of medical journals. Each journal consists of Bulgarian science and medicine topics and advises on how to tackle different medical issues that can occur.

From CmlMuSofia we have crawled 160 pages of medical journals. Each journal provides information about original scientific developments such as articles and reviews. Healthcare Management, Medical Ethics, and History of Medicine are also regularly covered in each journal.

From Bulsem we have crawled 1,924 pages of medical journals. Each journal has original articles from all fields of medicine and dentistry by Bulgarian and foreign authors.

From Basa we have crawled 1,353 pages of medical journals. Each journal consists of reviews, original articles, clinical cases, and case reports.

From MedSport we have crawled 1,793 pages of medical journals. Each journal covers problems of sports orthopedics, rehabilitation, physiology as well as the medical aspects of the training and competition process.

From Vma we have crawled 4,848 pages of medical journals. Each journal consists of scientific developments, publications from scientific medical forums, cases from the practice, and reports about new scientific events.

### 3.2 ICD-10 Classification Task Dataset

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Total Inst.</th>
<th>Unique Codes</th>
<th>Inst. w. Altern. Codes</th>
<th>Unique Tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>354,733</td>
<td>5,879</td>
<td>55,372</td>
<td>79,732</td>
</tr>
<tr>
<td>Train</td>
<td>284,144</td>
<td>5,879</td>
<td>-</td>
<td>76,909</td>
</tr>
<tr>
<td>Dev</td>
<td>35,117</td>
<td>5,876</td>
<td>26,186</td>
<td>31,753</td>
</tr>
<tr>
<td>Test</td>
<td>35,472</td>
<td>5,861</td>
<td>29,186</td>
<td>31,958</td>
</tr>
</tbody>
</table>

Table 2: ICD10 datasets: descriptive statistics for the number of alternatives codes.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Max</th>
<th>Mean</th>
<th>Median</th>
<th>Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>24</td>
<td>1.421</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Train</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Dev</td>
<td>22</td>
<td>1.409</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Test</td>
<td>24</td>
<td>1.431</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The ICD-10 classification contains several levels encoded with a different number of signs. The root level is encoded with the letters from the English alphabet, and subsequent levels append a number to the parent one. In this article we examine 3-sign and 4-sign codes, for example, the 3-sign A00 is the code for "Cholera", and 4-sign A00.0 is encoding a specific type of cholera - "Cholera due to Vibrio cholerae 01, biovar cholerae".

In the current article, we use the corpus published by Boytcheva et al (Boytcheva et al., 2020) as a basis and we perform additional preprocessing. It consists of two datasets: one with 189,756 3-sign samples and the other with 383,042 4-sign samples. The unique codes (classes) for each dataset are 2,035 and 10,971, respectively. It is important to emphasize that the dataset with 4-sign codes also includes 3-sign codes. The descriptions are in Bulgarian, Latin, and transliterated from Latin to Cyrillic. The second dataset (containing 4-sign and 3-sign codes) is used to process and conduct experiments with different BERT family models. ICD-10 codes are numerous, with some having only a few samples in the dataset. In other words, the dataset is highly imbalanced. For this reason, additional processing has been done, which aims to achieve three things:

1. Add artificially created samples. This is done by applying the following data augmentation techniques: word exchange; exchange of random letters in one word; delete any letter of a word; change any letter in a word to one close to it on the keyboard.

2. Codes that have less than 5 samples should be reduced to a higher level in the code hierarchy. This is possible because ICD-10 codes have a strictly specific hierarchy. For example, a 4-sign code like A00.0 has 4 levels - each of its symbols. The highest level is the letter. The next three levels are the numbers. About 4,939 classes in the dataset have less than 5 samples.

---

14https://github.com/BorisVelichkov/ICD10-Medical-Data
Those with a 4-sign code are reduced to their corresponding 3-sign code (remove the 4-sign specific class from the classification). 405 classes with 3-signs are under-represented and thus we cannot apply this approach for them.

3. To unite the classes that are not related to a particular disease but have a special purpose for capturing external factors influencing health. These are the codes V01-Y98 (External causes of morbidity and mortality) and Z00-Z99 (Factors influencing health status and contact with health services). They are reduced to the upper levels V and Z, respectively, following ICD-10 grouping logic.

The converted dataset (Full) is divided into three parts: train (Train), validation (Dev), and test (Test) datasets. An additional column for alternative codes has been added to the validation and test datasets, as a diagnosis can be assigned to more than one code. For each dataset, the number of samples, the number of unique codes, the number of samples with alternative codes, and the number of unique tokens are shown in Table 1. It is good to note that in the validation set there are 1,708 unique tokens that are not present in the training set, as well as 1,701 tokens in the test set that are not present in the training set. The intersection between these tokens is approximately one-third - 586 common unique tokens. Descriptive statistics such as the minimum number, the maximum number, the mean, and the median of the alternative codes are shown in Table 2. The numbers vary between 1 and 24, with most being closer to only one alternative code. An equivalent table with descriptive characteristics for the number of tokens is Table 3. There, the number of tokens varies between 1 and 34, but the average is between 4 and 5.

4 Deep Learning Methods for text-based classification

BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) is a deep learning language model pre-trained on a large corpus of data using bidirectional transformers that provides context-aware token and sentence representations. There are multiple BERT models for different languages and domains and BERT has shown very good results on a variety of different tasks. Transfer learning can be applied by using the published models and fine-tuning them with a smaller dataset on the target task.

We evaluate multiple BERT models by applying additional pre-training for Bulgarian medical texts using the masked language task and then fine-tuning them on the multi-class classification for ICD-10 codes.

For the masked language task, we mask the standard 15% of tokens and train the model to predict the correct token following the architecture from the original paper (Devlin et al., 2018). The goal of training is to minimize the perplexity of the model. We use the language model pre-training dataset to improve BERT’s understanding of Bulgarian medical text. We split the language model pre-training dataset in a proportion of 80:20 - 80% for training and 20% for testing.

WordPiece is used for tokenization and the original vocabulary from each model is used. As subword tokens are used, all words can be represented with tokens in the vocabulary. To train domain/language-specific extension to the vocabulary, a large corpus of training data would be required which is unavailable for Bulgarian.
the encoder and is trained to predict the correct ICD-10 class using a softmax activation. We return the top 5 classes with the highest probability as a prediction from the classifier as each diagnosis can belong to more than one class. We report accuracy, macro-F1, and mean reciprocal rank (MRR) metrics for the classification task.

The Multilingual BERT model uses BERT-base as a starting point and is additionally fine-tuned on the masked language task using Wikipedia articles in 104 languages incl. Bulgarian

BioBERT is based on BERT-base and fine-tuned on PubMed abstracts and PMC full-text articles

BlueBERT is a model based on BERT that is pre-trained on PubMed abstracts and (MIMIC-III) clinical notes

ClinicalBERT is initialized from BioBERT and trained on MIMIC-III, which contains around 2 million notes

SapBERT is a PubMedBERT that was further fine-tuned with synonym pairs from the knowledge base of UMLS, a collection of biomedical ontologies

SlavicBERT is a model, derived from MultilingualBERT, trained on Wikipedia articles in Bulgarian, Czech and Polish and news in Russian

5 Experiments and Results

Table 4: BERT fine-tuned models and their perplexity.

<table>
<thead>
<tr>
<th>BERT Model</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>BioBERT</td>
<td>1.7856</td>
</tr>
<tr>
<td>BlueBERT</td>
<td>1.8941</td>
</tr>
<tr>
<td>ClinicalBERT</td>
<td>1.7606</td>
</tr>
<tr>
<td>MultilingualBERT</td>
<td>3.2690</td>
</tr>
<tr>
<td>SapBERT</td>
<td>2.5644</td>
</tr>
<tr>
<td>SlavicBERT</td>
<td>5.6693</td>
</tr>
</tbody>
</table>

In the current article, experiments were performed with six different types of BERT models.

Each was additionally fine-tuned on medical articles in Bulgarian and then attached to the classification task to associate the diagnosis with the corresponding ICD-10 code. ClinicalBERT is fine-tuned for 20 epochs with a final perplexity of 1.7606 (the 12th epoch was 1.7788). SlavicBERT is fine-tuned for 16 epochs and has a perplexity of 5.6693 (the 12th epoch was 5.7312). All other models are trained in 12 epochs. All perplexities can be seen in the Table 4. In the classification task, all models are trained in 10 epochs. The change in loss can be seen in Fig. 2. Detailed results including Accuracy, Macro F1 and MRR are shown in Table 5. It is noted that the highest MRR and Macro-F1 is using MultilingualBERT (95% and 91%, respectively), followed by ClinicalBERT with 1% below (94%) MRR and 87% Macro-F1. ClinicalBERT has the highest accuracy - 92%.

As we can see all the models are doing quite well. What makes the ClinicalBERT one of the best is that this model is pre-trained on top of many clinical notes, which contain a large amount of medical concepts. Many of them are in Latin and are the same in their use in different languages. Also these notes are most likely quite close to medical diagnoses. It is also the model that has been fine-tuned for most epochs (20 epochs versus 16 for the...
Table 6: MultilingualBERT and BioBERT models Top 5 predictions for 3 diagnosis of real patients.

<table>
<thead>
<tr>
<th>Diagnosis Text</th>
<th>Multilingual BERT</th>
<th>BioBERT</th>
<th>True Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Захарен диабет 2 тип&quot; <em>(Type 2 diabetes mellitus.)</em></td>
<td>E11, E10, E12, P70.2, C91</td>
<td>E11, E10, E12, E13, N25.1</td>
<td>E11</td>
</tr>
<tr>
<td>&quot;Хронична лимфоцитна левкемия, B-клетъчна, IV к.с. по Rai&quot; <em>(Chronic lymphocytic leukemia, B-cell, IV hp according to Rai)</em></td>
<td>C91, C91.1, C91.0, C83.5, C83</td>
<td>C91, C91.1, C91.0, C83.5, C94</td>
<td>C91.1</td>
</tr>
<tr>
<td>&quot;Хронична лимфоцитна левкемия – B-кл., CD5+, III ст. по Rai, «C» по Binnet CIRS score-16&quot; <em>(Chronic lymphocytic leukemia - B-class, CD5+, III st. by Rai, “C” by Binnet CIRS score-16.)</em></td>
<td>C94, C94.7, C88, C88.0, C91</td>
<td>C91, C91.0, C91.1, C83.5, C83</td>
<td>C91.1</td>
</tr>
</tbody>
</table>

SlavicBERT and 12 for the rest). This may be the reason why ClinicalBERT leads MultilingualBERT in the accuracy (with 5% better). MultilingualBERT, on the other hand, is trained in over 100 languages, which may allow it to do very well in different languages and to be relatively easy to be fine-tuned on new data. Similarly, it can be said that another advantage is that the diagnoses combine text in Bulgarian, Latin and transliterated from Latin to Cyrillic. In addition, we can say that in other studies for the same task in Bulgarian, MultilingualBERT is the model that gives the best results. Here it has best macro F1 and MRR.

In order to be able to illustrate in a more understandable way the task we will show three examples of real diagnoses from discharge letters of patients and how two of the classifiers (MultilingualBERT and BioBERT) predicted codes of these diagnoses in the Table 6. As we can see the two classifiers return the true code at first position for the first diagnose. Also for this sample BioBERT has more close predictions in the top 5. The second and the third diagnoses are a little bit more complex because they have 4-sign code which in these cases is same for both - "C91.1". As can be seen, in addition to the same code, the two diagnoses are very similar in text. For both diagnoses, BioBERT returns the 3-sign code first and the exact 4-sign code second. The same thing is seen with MultilingualBERT, but only for the first of the two diagnoses. For the second, the results are worse and the classifier can only guess the 3-sign code of the diagnosis. An interesting observation is that it is also in fifth place in the top 5 predicted codes.

In contrast with the results presented in (Velichkov et al., 2020) the SlavicBERT model in our experiments shows comparable results with the other models, and moreover it is the second ranked model on the basis of accuracy. Another difference in our results is that ClinicalBERT outperforms BioBERT in all three evaluation metrics - accuracy, macro F1 and MRR. In both cases the reason is longer training (more epochs) for the fine-tuning of the models.

6 Conclusion

In the current article, a comparative analysis of six different BERT models is made, each of which is trained on a large amount of data and additionally fine-tuned on a big corpus of medical texts in Bulgarian. It can be said that the selected models are a good representative sample for the task on which they are applied, as among them there are models trained on over the top 100 languages, trained on Slavic languages, trained on medical and bio literature. The results obtained are quite high and show that all tested models are promising. As future improvements, it would be good for all models to be fine-tuned further, both with more texts and with more epochs. It would be good to equalize the number of epochs of fine-tuning for all models. Also other good improvements would be comparing the models before and after fine-tuning and applying cross-validation to more accurately evaluate the models.

Acknowledgments

This research is funded by the Bulgarian Ministry of Education and Science, grant DO1-200/2018 ‘Electronic health care in Bulgaria’ (e-Zdrave).

Also funded via GATE project by the EU Horizon 2020 WIDESPREAD-2018-2020 TEAMING Phase 2 under GA No. 857155 and OP SE4SG under GA No. BG05M2OP001-1.003-0002-C01.
References


Mistake Captioning: A Machine Learning Approach For Detecting Mistakes and Generating Instructive Feedback

Anton Vinogradov
University of Kentucky
Lexington, KY 40506
anton.vinogradov@uky.edu

Andrew Miles Byrd
University of Kentucky
Lexington, KY 40506
andrewbyrd@uky.edu

Brent Harrison
University of Kentucky
Lexington, KY 40506
harrison@cs.uky.edu

Abstract

Giving feedback to students is not just about marking their answers as correct or incorrect, but also finding mistakes in their thought process that led them to that incorrect answer. In this paper, we introduce a machine learning technique for mistake captioning, a task that attempts to identify mistakes and provide feedback meant to help learners correct these mistakes. We do this by training a sequence-to-sequence network to generate this feedback based on domain experts. To evaluate this system, we explore how it can be used on a Linguistics assignment studying Grimm’s Law. We show that our approach generates feedback that outperforms a baseline on a set of automated NLP metrics. In addition, we perform a series of case studies in which we examine successful and unsuccessful system outputs.

1 Introduction

Giving feedback is one of the most critical parts of education and training. It allows the instructor to correct errors in students’ understanding and guide them to the correct solution. In automated learning systems, such as intelligent tutoring systems (ITSs) the feedback and hints are focused on getting the student to complete the assignment, but this may not take into account their mastery of that skill. While better feedback can be made by considering student intent, we find that current models are lacking in this regard. They are typically focused on learning objective mastery and may not necessarily give the whole picture when determining why students answered something the way they did.

But there is a natural solution to this, in that skilled educators are capable of intuitively modeling a student’s thought process and determining mistakes in it. This skill is often developed over time. The difficulty of this task for educators depends on the field of study, with some more free form fields like writing and programming being particularly difficult. By indirectly modeling a student’s intent by instead modeling how the instructor gives feedback, we can use this model to generate novel feedback sentences.

We propose a system that when given a mistake that has been identified in the task of deriving sound changes from Proto-Indo-European to Proto-Germanic, will automatically generate an explanation that identifies the mistake and the reasoning to why this sound change should not apply in this case. We call this method Mistake Captioning as it takes inspiration from image captioning methods and post-hoc AI rationale generation methods such as Ehsan et al. (2019) Automated Rationale Generation.

We apply a suite of automated machine translation and image captioning metrics to test our system. We also take a closer look at 3 cases from a more complete retraining of the network to better understand how the network is performing and how well the metrics perform regarding our task. We find that this method is a good start, but more data and work need to be done to be able to integrate this into a serious learning environment.

2 Background and Related Work

In the space of ITSs a problem is how to get a student to the correct solution. One of the ways to achieve this is by having an expert to author the correct solution and the path to take to get there (Marwan et al., 2020; Unnam et al., 2019; Ariely et al., 2020). These all require the use of an expert to either structure the assignment such that feedback can be extracted or to label assignments to create a system that can learn the expert knowledge. In some domains this is not necessary as the students can provide the data for the path to the solution themselves. This data-driven approach
uses data from multiple students to create all possible paths that lead to a solution, as shown by Stamper et al. (2008) in their work on Hint Factory. This was later extended to create paths that optimize for productivity metrics by Maniktala and Barnes (2020). In vast solution spaces generating all possible paths to the solution is not feasible, and in Rivers and Koedinger (2017) work this is done by abstracting states so that a single state can represent a large space of states.

Our method does not attempt to establish a correct answer like these previous methods as they do not tackle an underlying problem as to why students make mistakes. We structure our problem not as a method of guiding the student to the correct path, but by trying to correct the thought process that led them to their current answer. This is similar to the methods used in image captioning (You et al., 2016) where they train a model to recognize what is in an image by relating human authored captions to images. This model is then used to generate captions on similar features in new pictures. This is also like the work done in post-hoc AI rationale generation like that of Ehsan et al. (2019). In this work they relate a set of human actions to human rationales of those actions for the use of explaining AI behavior. In our system, we make use of human actions in the form of mistakes and relate them to human explanations of the mistake.

2.1 The Proto-Indo-European Language and Grimm’s Law

Our application area for this paper is on an assignment concerning Grimm’s Law, which is a series of sound changes that occurred in the evolution of Proto-Indo-European (PIE) into Proto-Germanic (PGmc). PIE is a reconstructed language that attempts to recreate the common ancestor of Indo-European language family. While no direct evidence remains of PIE, the similarities between the languages in this family indicate that they come from a common ancestor. Despite these similarities, there are still differences in the languages that can be attributed to shifts in pronunciation as time went on and communities speaking this language became isolated from others. PGmc. is also a reconstructed language, serving as the source of all Germanic languages. Sound changes can be traced from PIE to PGmc.

We focus our task on a set of sound laws that describe how stop consonants change from PIE to PGmc collectively known as Grimm’s Law (Campbell, 2013) and also include instances where Grimm’s Law does not occur. Grimm’s Law consists of three different changes that all contribute towards a single shift known as a chain. These changes are as follows:

1. PIE voiceless stops change to voiceless fricatives.
2. PIE voiced stops change to voiceless stops.
3. PIE voiced aspirated stops change to voiced stops or fricatives.

Together they are chained such that affected sounds in PIE are shifted one step to their form in PGmc. This limited set of shifts is what is represented in our data and is the reason for some of the limitations.

3 Methods

3.1 Data

The data used in this paper is centered around the linguistics PIE rule, Grimm’s Law (Campbell, 2013). Since the task is to generate a reasoning for a mistake made by students, we have opted to organize the data as a fill-in-the-blank task, as one would show up as an assignment. Since we are focusing on captioning the mistake, the changes from PIE to PGmc must contain a mistake and an explanation for that mistake. To create these erroneous responses, we had a linguistics expert generate a number of entries fulfilling these categories: a PIE word; a PGmc form of that word with blanked areas demarked by underscores; a faulty PGmc form containing only a single mistake each, so that the explanations can be more focused on that specific mistake. When multiple mistakes are identified for each PIE word, they are separate entries with separate explanations. Likewise, if there are multiple blanks to fill in for a word, then each blank will have a separate set of entries for its possible mistakes. The explanations were written with a simplified style as to not introduce too much detail into the explanations that
PIE | Blanked PGmc | Faulty PGmc | Explanation
--- | --- | --- | ---
pis'kós | is az | fishaz | k ultimately does become h, but not when immediately following stops & fricatives
pis'kós | is az | biskaz | p can shift to b, but only when the middle of the word preceding an accented vowel
pis'kós | is az | fisgaz | k can shift to g, but only when preceding an accented vowel in the middle of a word
pis'kós | is az | fisxaz | k normally changes to x, but not when preceded by a stop or fricative

Table 1: The data entries for the mistaken sound changes in the PIE word for “fish”

resulted in each explanation being a single sentence following a general format. This sentence is in the rough form of “the mistake that was made, reason why it is not applicable in this case”. For example, in Table 1 the explanation for why fishaz is wrong has k ultimately does become h as the mistake that was made and but not when immediately following stops & fricatives as the reason why it is not applicable in this case.

This data was gathered by a single Linguistics Expert, with minor input from the authors as to style and structure to better suit this data for machine learning. Our linguistics expert is an associate professor of linguistics that frequently teaches classes on PIE. There were no attempts to control for consistency in tone, tense, or voice style, so such variations do occur in the data. Because the sentences follow a general format, there are some exact explanation matches in the data. Since this was the application of a small set of rules over a larger set of words, finding common letter changes and explanations could not be avoided. We collected 163 entries of PIE words, fill-in-the-blanks, faulty PGmc words, and their corresponding explanations. These came from 55 unique PIE words and contained 53 unique explanations to cover all cases found in the data. The most common explanation occurred 15 times, while others only occurred once in the entire dataset. Even those that occurred once often shared similar phrasing with other explanations since they were drawn from a common set of rules and a common set of consonants.

The data was further structured for training as seen in Table 2. To preserve the question that was being hypothetically asked to solve and the answer, we combined the PIE form and the faulty PGmc form, separated by a space, as our model input. For our model output we used the explanations without adding anything, but we did remove and change some characters. We removed all instances of ending punctuation like periods and exclamation points and spaced out commas to count them as separate tokens. The white space was also normalized and any leading or trailing white space was removed. While not shown in the table, the input was tokenized character by character to help preserve the differences between the faulty PGmc, while the output was done word by word.

3.1.1 Data Noise

Our data is very limited in scope due to the effort it requires to generate it. If we were to train our model only using the data as it is, we would likely run into issues with overfitting and overtraining. To alleviate this, we have injected noise into the data to be able to train longer. For our training output, creating noise is a straightforward process. In every training iteration, the output has 30% of its words masked out. This value was chosen to keep most of the explanation intact while still having a significant amount noised as overfitting is a serious concern. This means that at no training iteration has the model ever seen a complete output sentence, but with enough training iterations the unmasked portions should have overlapped enough to reveal the complete sentence.

The input was a more complicated process. Since the input is smaller than the output, we made the attempt to preserve some of the important parts, namely the actual changes from PIE to PGmc. Since these are fill-in-the-blank tasks, we have access to the parts that contain no mistakes and using this part of the dataset we identified the portions
of the Germanic words that are considered safe to mask out in the form of maskable indices. These are shown in the 3rd column of Table 2, though these may not visually line up to the index of the words since letters that are accented are encoded as multiple characters. Since this only affected the PGmc part of the already smaller input, instead of masking 30% of the characters we only mask a single character at a time and limit it to only 30% of the time. This hopefully allows us to make a more robust set of training examples without disrupting the underlying meaning too much.

### Table 2: The processed data entries for the word “fish”.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Maskable Indices</th>
</tr>
</thead>
<tbody>
<tr>
<td>piskós fishaz</td>
<td>ₖ ultimately does become h , but not when immediately following stops &amp; fricatives</td>
<td>[10, 11, 13, 14]</td>
</tr>
<tr>
<td>piskós biskaz</td>
<td>p can shift to b , but only when the middle of the word preceding an accented vowel</td>
<td>[10, 11, 13, 14]</td>
</tr>
<tr>
<td>piskós fisgaz</td>
<td>ₖ can shift to g , but only when preceding an accented vowel in the middle of a word</td>
<td>[10, 11, 13, 14]</td>
</tr>
<tr>
<td>piskós fisxaz</td>
<td>k normally changes to x , but not when preceded by a stop or fricative</td>
<td>[10, 11, 13, 14]</td>
</tr>
</tbody>
</table>

3.2 Model

For our model we use a sequence-to-sequence network. Sequence to sequence networks utilize two recurrent neural networks: an encoder and a decoder. The encoder encodes the sequential data into a fixed length context vector that is meant to represent the important elements of the input. This context vector is then used by the decoder to generate a natural language output. In our case, this output is an explanation, but this can be used for other tasks. Sequence to sequence networks have been used for translation (Sutskever et al., 2014) and post-hoc rationale generation (Ehsan et al., 2019) tasks, and while our task is not a traditional translation or rationale generation task it can be modeled as one. The network learns to take the input sequence of our PIE and faulty PGmc words as a character sequence and translate them into a word sequence for our explanations. By connecting the input to the explanation, we are learning human-like reasoning for why the mistakes happened. We use Gated Recurrent Units (GRUs) to take advantage of their faster training time and better use with smaller data (Chung et al., 2014). We also make use of an attention mechanism (Graves, 2013) on the decoder to learn to focus on the important parts of the output explanation.

The model was trained on 75,000 iterations using the same 163 question and explanation pairs, but as mentioned previously in each iteration some characters and words were masked to prevent the model from seeing them. During each training iteration a random sample from the data is run through the network after having noise applied.

4 Experiments

We conducted two experiments to evaluate our method: A quantitative approach using a set of automated metrics to judge the candidate explanations, and a qualitative case study to demonstrate what types of outputs were generated. For both the qualitative and quantitative evaluation we used 5 different automated metrics: BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), ROUGE (Lin, 2004), CIDEr (Vedantam et al., 2015), and BERTScore (Zhang et al., 2019).

For our quantitative evaluation, we used k-fold cross-validation to control for the variance in our data. We compare our approach against a majority baseline. Since several of the explanations in the data were repeated, we decided that a better baseline would be the majority baseline and used the most repeated explanation in our data. This explanation was used 15 times in the data, and for the majority baseline we set the generated output to all be this exact sentence. This explanation can be seen in Table 4 in the baseline incorrect, candidate sentence. While the majority baseline has a low
accuracy, about 9.2%, it was chosen to show that the method is viable before continuing to a human evaluation.

For our qualitative analysis, we opted to look at outputs for a retrained model that was trained on all but 8 random examples of the data. Since our dataset is small, we decided that training on all but a small number of examples would give better insights on what the network was learning. We examined how the model performed in generating explanations for each of the 8 examples that were held out during training. From these 8, we used their automated metric performance in conjunction with their correctness to divide them into three categories: correct, partially incorrect, and incorrect. We then explore why this occurred and reason about the implications of these results in practice.

During training, we use the following hyperparameter values for our model. We used a teacher forcing ratio of 0.5, and a learning rate of 0.01. For the hidden layers (consisting of the attention and GRU layers), a fixed size of 256 was used. All outputs were limited to a maximum length of 30 but this limit was not reached. To test the model, we implemented beam search to select the best candidate explanation instead of the default greedy search. In addition to improving the accuracy of the model this allowed us to look at multiple final outputs, each scored by the model. This is useful in manually judging if the alternatives to the top output were more correct. We limited the beam width to 5.

5 Results

We found that our method consistently outperformed the baseline in all metrics as can be seen in Table 3. This provides evidence that our model is working to learn when to apply and how to recreate humanlike feedback in response to mistakes. We also found that the leave-8 retraining performed better than the 5-fold models. This trend can be visually seen in Figure 1.

Applying a 2-sided t-test we find that the difference between the baseline and the 5-fold is statistically significant ($p < 0.0001$) in all cases. When comparing the baseline to 8 case studies this also holds true ($p < 0.001$). There is also no statistically significant difference between the case studies and the 5-fold.

6 Discussion

In this section, we discuss the results in more detail, going over the results of the automated metrics and case studies.

6.1 Automated Metrics

While the automated metrics do show that our method outperforms our baseline, they are not perfect. One finding that came out of this evaluation is that automated metrics can be misleading when evaluating performance. This is because automated metrics for evaluating language generation systems often measure how well generated sentences overlap with reference sentences. The change of a single word alters the semantic meaning of an explanation, however, which can fool the metrics into scoring it higher than it is. Likewise, a generated explanation can be nearly correct but contain little similarity in structure to the original. This is ultimately a limitation of the data since we only have a single reference explanation for each faulty PGmc example. In the data there exist explanations that explain a concept similar to other explanations but are differently structured. If the model lifts the structure of one explanation to the correct change and reasoning of a change, there is no guarantee that this will match the reference explanation and will be scored more poorly. This does not mean that these metrics are not useful, though, just that the limited scope of our data makes it less suited to these metrics. There is a general trend between how well the explanation performs compared to how correct it is, as we will see in the case studies. In general, this seems to suggest that how
Baseline 5-Fold Case Studies

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Std.</th>
<th>Mean</th>
<th>Std.</th>
<th>Mean</th>
<th>Std.</th>
</tr>
</thead>
<tbody>
<tr>
<td>bertscore</td>
<td>0.8896</td>
<td>0.0397</td>
<td>0.9479</td>
<td>0.0502</td>
<td>0.9593</td>
<td>0.0521</td>
</tr>
<tr>
<td>Bleu_1</td>
<td>0.2872</td>
<td>0.2424</td>
<td>0.6019</td>
<td>0.2690</td>
<td>0.6776</td>
<td>0.2999</td>
</tr>
<tr>
<td>Bleu_2</td>
<td>0.2377</td>
<td>0.2570</td>
<td>0.5558</td>
<td>0.3009</td>
<td>0.6503</td>
<td>0.3202</td>
</tr>
<tr>
<td>Bleu_3</td>
<td>0.2059</td>
<td>0.2633</td>
<td>0.5228</td>
<td>0.3198</td>
<td>0.6322</td>
<td>0.3239</td>
</tr>
<tr>
<td>Bleu_4</td>
<td>0.1802</td>
<td>0.2676</td>
<td>0.4913</td>
<td>0.3378</td>
<td>0.6180</td>
<td>0.3266</td>
</tr>
<tr>
<td>METEOR</td>
<td>0.2175</td>
<td>0.1749</td>
<td>0.4003</td>
<td>0.1908</td>
<td>0.4532</td>
<td>0.1917</td>
</tr>
<tr>
<td>ROUGE_L</td>
<td>0.3377</td>
<td>0.2709</td>
<td>0.6495</td>
<td>0.2864</td>
<td>0.7481</td>
<td>0.3112</td>
</tr>
</tbody>
</table>

Table 3: Mean and Std. for all models generated.

well these metrics will perform will change if we get more data, both on the number of PIE words and the number of possible explanations for each mistake.

6.2 Case Studies

To take a better look at the generated explanations we have a separate leave-8-out training. In these 8 that we left out and tested on we found three categories: correct, partially incorrect, and completely incorrect. Correct explanations include the correct mistaken change and the correct reason it was not applicable, while partially incorrect only include one of the two. Incorrect explanations contain neither. Of the 8, 4 of these where correct, 3 were partially incorrect, and only one was completely incorrect. We have selected 3 examples to match these 3 categories to look more closely at.

6.2.1 Completely Incorrect

There was only a single example of the completely incorrect case, which is shown in Incorrect case in Table 4. This explanation does not identify the sound change giving \( k \) instead of \( t \) for the sound change. For the reasoning it seems like it is on the correct track by mentioning \textit{preceded by stop or fricative} but reverses whether the change occurs because of this. In fact, this is a case of the candidate explanation having no relation to the reference explanation, nor is close to any reference explanation for that PIE word, nor the PIE and PGmc input. The candidate explanation refers to a \( k \) which does not appear in input but, as will be expanded on later, this may be a case of encoding. Due to the way that the input is interpreted on a character-by-character basis the model may have recognized the presence of the \( k \) and conflated it with a \( k \). It also used the explanation that is most repeated in the dataset, which suggests that it picked the most popular explanation as opposed to correctly learning this change.

6.2.2 Partially Incorrect

For the partially incorrect case we find that the model could identify something correct but not everything. Sometimes this was only identifying the correct sound change or the correct reasoning, but in the case that we are focusing it generated a completely new explanation that was one word from being correct. This candidate explanation changed the word \textit{must to only} which completely changes the meaning of the explanation. In our explanation generation we saved the top 5 sentences, and the correct explanation does appear as the second. This may be another encoding issue since the phrase \( k \textit{only} \) appears several times in our dataset. We used beam search to attempt to avoid this problem, but in this case, it does not seem to have been enough. The complete phrase that was generated does not appear as a reference explanation in any of the examples, so it is unknown as to why it chose this one. It is likely that having more reference explanations would help as it would force the model to learn more important features.

In another case of partially incorrect, we found that the exact reference explanation that never appeared in that training data, which means that the model could not have possibly generated a completely matching explanation.

6.2.3 Correct

The correct category is marked by when the model was correctly able to generate an explanation for the mistake that was made, matching both the change and reasoning. This is also the largest category, comprising 4 of the 8 examples. These score highly on all the metrics. This category has sound changes that are well represented in the training data, with no reference explanation appearing less than 5 times. In the case shown in Table 4, the
Table 4: Specific examples of cases that we found in our data. Included is an example from the baseline.

<table>
<thead>
<tr>
<th>Case</th>
<th>PIE PGmc</th>
<th>Reference Sentence</th>
<th>Candidate Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completely</td>
<td>skotó</td>
<td>t only remains t if it’s preceded by a stop or fricative</td>
<td>while k does become a velar sound, it also shifts if it isn’t preceded by a stop or fricative</td>
</tr>
<tr>
<td>Incorrect</td>
<td>skata</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partially</td>
<td>ñlew-</td>
<td>k̂ must shift if it isn’t preceded by a stop or fricative</td>
<td>k̂ only shift if it isn’t preceded by a stop or fricative</td>
</tr>
<tr>
<td>Incorrect</td>
<td>ñlew</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Correct</td>
<td>pénkʷe</td>
<td>kʷ only remains kʷ if it’s preceded by a stop or fricative</td>
<td>kʷ only remains kʷ if it’s preceded by a stop or fricative</td>
</tr>
<tr>
<td></td>
<td>finkʷe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Baseline</td>
<td>h₂yuh₂ñk̂-</td>
<td>ultimately k becomes h, but it first changes into a velar sound</td>
<td>while k does become a velar sound, it also shifts if it isn’t preceded by a stop or fricative</td>
</tr>
<tr>
<td>Incorrect</td>
<td>yunha-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

reference explanation appeared 8 times which is relatively high, but not unique to this category. The completely incorrect case had its reference sentence appear 12 times in the data, meaning that this is likely not the most prominent factor in whether a generated explanation will be correct or incorrect.

In all these cases we find that the automated metrics to judge these explanations are generally higher when the answer is more correct, and generally lower when it is more incorrect. This holds in cases when there is only partial correctness. As mentioned before this is likely due to our limited data and may break if we have more reference sentences. We hope to expand the data to cover other changes in linguistics, though the method should be able to be transferring to other domains too.

7 Future Work

Due to the way that our method works in identifying a single mistake at a time, we can use the fill-in-the-blank task to generate explanations on multiple mistakes. This is done by first comparing the answer given to the correct answer and finding the difference between the two and isolating each individual mistake. Each of these individual mistakes can be then applied to the original correct answer to create separate faulty answers which can subsequently be run through our system. Using this method, we can also change the fill-in-the-blank task to a simple response, where the student is tasked to correctly make all the changes to the PIE word to produce a PGmc word.

Either through separate models for separate rules (and enough annotations on the input) or by training a single more complex model, it may be possible to create a system that can caption all mistakes for a given task. This could work in conjunction with automatically grading the assignments to quickly provide feedback on the mistakes.

8 Conclusion

In this paper we show a method of the novel task of creating automated captions of mistakes. These captions serve as explanations for what the thought process behind the mistake was, and why it is not applicable in this case. We apply two methods to test our model, comparing against a baseline with a suite of automated metrics, and manually identifying and analyzing the sentences in a series of case studies. Our experiments show that our method has promise in creating these automated captions but that there are significant challenges that need to be overcome.

We set out tasks in the domain of Linguistics pedagogy, and in this we can make improvements with representation of the input data and with gathering more and more varied data. We also hope to expand this to include a wider set of rules covering a larger set of sound changes, and eventually evaluate the method using a human study. We hope that this method of mistake captioning can be applied to other fields of study.
References


Abstract

Post processing is the most conventional approach for correcting errors that are caused by Optical Character Recognition (OCR) systems. Two steps are usually taken to correct OCR errors: detection and corrections. For the first task, supervised machine learning methods have shown state-of-the-art performances. Previously proposed approaches have focused most prominently on combining lexical, contextual and statistical features for detecting errors. In this study, we report a novel system to error detection which is based merely on the n-gram counts of a candidate token. In addition to being simple and computationally less expensive, our proposed system beats previous systems reported in the ICDAR2019 competition on OCR-error detection with notable margins. We achieved state-of-the-art F1-scores for eight out of the ten involved European languages. The maximum improvement is for Spanish which improved from 0.69 to 0.90, and the minimum for Polish from 0.82 to 0.84.

1 Introduction

Post processing is the most conventional approach for correcting errors that are caused by Optical Character Recognition (OCR) systems. Traditionally, the task is divided into two subtasks: (1) Error detection to classify words as either erroneous or valid, and (2) Error correction to find suitable candidates to correct the erroneous words (Kolak and Resnik, 2005; Kissos and Dershowitz, 2016; Mei et al., 2016). A large body of work has proven the success of statistical and supervised machine learning methods for both subtasks (Afli et al., 2016; Schulz and Kuhn, 2017; Nguyen et al., 2018; Amrhein and Clematide, 2018; Nguyen et al., 2019).

Machine learning methods largely rely on feature engineering for their performances, particularly in supervised settings. Feature engineering involves exploring various features and feature combinations that best characterise the data. However, for post-OCR error detection, finding a suitable set of features is challenging because of the diversity of OCR errors (Amrhein and Clematide, 2018). This has been demonstrated in previous work and more recently in the ICDAR competitions (Chiron et al., 2017; Rigaud et al., 2019) where various features have been explored with varying success rates (more details in Section 2).

To address this challenge we propose a novel approach to the error detection task. Instead of examining more features we focus merely on a single feature, namely the n-gram counts of the candidate token. Our approach is inspired by dictionary lookup approaches, which are known for their simplicity and efficiency but are restricted to the dictionary size. Since building large-scale dictionaries is a challenging task in itself, we propose to generate n-grams of a given candidate token, and then use their counts as the only feature to train machine learning models (more details in Section 3). We evaluate our method on the ICDAR2019 dataset (Chiron et al., 2017) and compare the results to a number of approaches reported in the competition (Section 4). Our system achieves state-of-the-art results for eight out of the ten involved languages by beating the previous results with fair margins and comparable scores for the remaining two languages. Our approach is very simple and is computationally less expensive as it does not require any other feature computation apart from the n-gram counts.2

2The data and models are available under CC BY
2 Related Work

Our work takes inspiration from dictionary based approaches and uses supervised machine learning techniques to train a post-OCR error detection model. We therefore focus on reviewing the works related to dictionary and statistical supervised machine learning based approaches.

Simple dictionary based approaches have performed reasonably well for various natural language processing tasks (e.g. Hull and Grefenstette (1996); Sindhu and Sagar (2017)). It is therefore not surprising the approach has been applied to detecting OCR errors before (Schulz and Kuhn, 2017; Nguyen et al., 2018). Taking this approach, dictionaries and large word lists are usually compiled from corpora and other sources. Each token in the data is then compared with the word in the dictionary to determine whether it is an error word or not. This approach has been explored by the CSITJ team who was among the six successful teams participating in the ICDAR 2019 competition (Rigaud et al., 2019). The team compiled a dictionary of over 370 thousand English and French words and checked each word in the dataset against it.

Dictionary lookup methods for post-OCR are challenging because they usually suffer from out-of-vocabulary problem. Another limitation is in detecting real-word errors, i.e. the word appears in the dictionary but is wrong in its context. Therefore alternative methods to OCR error detection have been proposed (see a comprehensive survey of existing methods by Nguyen et al. (2021)).

The remaining teams in ICDAR 2019 applied techniques from: (i) Context-based character correction using BERT (CCC) – the winning system; (ii) Character level attention approach using the open source system OpenNMT (CLAM); (iii) Weighted finite-state transducers based on noisy channel model (REA1&2); and (iv) Character level seq2seq multi-layer LSTM (UVA).

Other approaches to post-OCR error detection combined character-, word-n-grams and context based features to train a machine learning model (Mei et al., 2016; Khirbat, 2017; Nguyen et al., 2019; Dannéls and Persson, 2020). Mei et al. (2016) trained a regression model on 6 features containing n-gram and context information. Khirbat (2017) trained a support vector machine (SVM) model with 3 features: presence of alphanumeric characters, bi-gram frequency of the word and context information that is if the word appears with its context in other places. Nguyen et al. (2019) trained a Gradient Tree Boosting classifier on a set of 13 character and word features on two datasets of English historical handwritten documents (monograph and periodical) taken from the ICDAR competition (Chiron et al., 2017). The features they experimented with include character and word n-gram frequencies, part-of-speech, and the frequency of the OCR token. Dannéls and Persson (2020) trained an SVM model on 6 statistical and word based features including the number of non-alphanumeric characters, number of vowels, word length, tri-gram character frequencies, number of uppercase characters and the amount of numbers occurring in the word.

As many authors point out the choice of the features is essential for the performance of the machine learning model. The advantage of these methods is that they are trained to detect both real-word and non-word errors. The drawback is they require laborious feature engineering.

Laborious feature engineering is a bottleneck not only for machine learning but also for other statistical approaches that rely on pre-defined features extracted from data, such as noisy channel approaches (Evershed and Fitch, 2014; Kissos and Dershowitz, 2016; Drobac et al., 2017).

3 Method

3.1 Datasets and Preprocessing

We used the datasets from the ICDAR2019 competition on post OCR error detection and correction. The total size of the original data is 22 million characters and it contains varying numbers of characters for ten European languages (Bulgarian, Czech, German, English, Spanish, Finnish, French, Dutch, Polish, Slovenian) together with the corresponding ground truth data. The dataset comes with the OCRed and ground truth aligned at the character level. For our experiments, we needed to align it at the token (word) level. We did that by tokenizing the ground truth at space and for each token taking the same number of characters from the OCRed version. After we removed the special alignment symbols (‘@’ and ‘#’) inserted by organizers for alignment. The resulting OCRed and ground truth tokens were compared to set the labels ‘0’ if the

4.0 licence at https://github.com/spraakbanken/NovelOCRErrorCorrection.
token was erroneous or ‘1’ if the token was valid. These labels are the dependent variables that are to be learned and predicted by the machine learning models. Table 1 shows example tokens, ground truth, and the labels from English, Danish, and Finnish datasets. Table 2 shows the number of training and test sets produced for each language after removing the duplicates and instances with ‘NA’ values and the class percentage i.e. what percentage of the total is the class ‘0’ and ‘1’ in the training and test sets.

### 3.2 Machine Learning Models and Encoding

Machine learning classifiers are known to have pros and cons depending on the task at hand. Dannélls and Virk (2020) compared between 5 state-of-the-art machine learning classifiers including Logistic Regression, Decision Tree, Bernoulli Naive Bayes, Naive Bayes and SVM. They found that SVM is the best choice for post-OCR detection. Others have also shown the performance of SVM is equivalent to the performance of artificial neural networks (Arora et al., 2010; Hamid and Sjarif, 2017; Amrhein and Clematide, 2018). In response to these previous experiments, in this study we have chosen to experiment with SVM models.

We take advantage of the implementation of the machine learning algorithms in the `sklearn` module (Pedregosa et al., 2011). Because the module requires the data to be in numeric form we used one-hot encoding for data transformations (see Section 4.1). While the details of the encoding method are beyond the scope of this paper, the major idea behind one-hot encoding is to add an extra dimension in the feature vector for each unique feature value. This produces an N dimensional feature vector (the learned encoding), where N is the total number of unique values of all features. In our case we have words as the training data. Suppose \([w_1, w_2, w_3, \ldots, w_n] \) is the set of unique words, and \([0, 1, 1, \ldots, 0] \) is the set of corresponding labels representing whether the word is erroneous or not. The learned one-hot encoding will be a \((n+2)\) dimensional vector, where \(n\) is the unique number of words in the training data and there are 2 unique label values. Each word is then encoded by setting the corresponding word and label dimensions of the vector to 1 while the remaining dimensions are set to 0.

### 3.3 Generating the Machine Learning Features

As mentioned previously, we took inspiration from dictionary look up based approaches, but in this study we have applied it in a novel way. Instead of building a dictionary separately from different external resources, we let our SVM model build it from the training data. This was achieved by learning one-hot encoding from the training data (i.e. words), encoding the training data and then using the resulting vectors as the only feature to train the SVM model. In other words, we turn the model into a dictionary lookup kind of system as the model memorizes vectors of each training instance. During prediction the trained model simply relies on the observed word, i.e. whether its encoded feature vector has been seen in the training data and predicts accordingly.

This type of approach has a major restriction that it is not scalable and is bound to feature values seen in the training data. In our case this means if a word has not been seen in the training data the system will simply fail to predict whether it is erroneous or not. Another downfall is that depending on the size of training data it may take days to train such models. To overcome these limitations we experimented further with the n-gram approach. Instead of using the complete word, for each candidate token, we generated character uni-grams, bi-grams, and tri-grams from it. These n-grams together with their counts within the token were used as feature values to train and test the model. To take an example, suppose our candidate word is ‘passenger’. The computed uni-, bi-, and tri-gram counts will be as follows:

- **uni-gram** \{'a':1, 'e':2, 'g':1, 'n':1, 'p':1, 'r':1, 's':2\}
- **bi-gram** \{'p':1, 'as':1, 'en':1, 'er':1, 'ge':1, 'ng':1, 'pa':1, 'r':1, 'se':1, 'ss':1\}
- **tri-gram** \{'pa':1, 'ass':1, 'eng':1, 'er':1, 'ger':1, 'nge':1, 'pas':1, 'sen':1, 'sse':1\}

It is worth mentioning that the scope of n-gram counts is limited to the word itself, rather than the entire training data i.e. these counts represent the occurrence of a particular uni-, bi-, or tri-gram within the word rather than the total count of the n-gram in the training data. Previous authors have attempted to exploit n-gram frequencies (e.g. Mei et al. (2016); Khirbat (2017)) computed over the
Table 1: A sample from the English, Danish, and Finnish datasets after the preprocessing step (GT = Ground Truth).

<table>
<thead>
<tr>
<th>English</th>
<th>Danish</th>
<th>Finnish</th>
</tr>
</thead>
<tbody>
<tr>
<td>matter</td>
<td>matter</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>the</td>
<td>1</td>
</tr>
<tr>
<td>king@</td>
<td>king @</td>
<td>0</td>
</tr>
<tr>
<td>very</td>
<td>very</td>
<td>1</td>
</tr>
<tr>
<td>glad</td>
<td>glad</td>
<td>1</td>
</tr>
<tr>
<td>hereof @</td>
<td>hereof</td>
<td>0</td>
</tr>
<tr>
<td>@Hewise</td>
<td>likewise</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2: Training and test dataset statistics

<table>
<thead>
<tr>
<th>Language</th>
<th>Training Set</th>
<th>Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>#words</td>
<td>%</td>
</tr>
<tr>
<td>Bulgarian (BG)</td>
<td>17844</td>
<td>0.37</td>
</tr>
<tr>
<td>Czech (CZ)</td>
<td>7227</td>
<td>0.19</td>
</tr>
<tr>
<td>Danish (DE)</td>
<td>18216</td>
<td>0.29</td>
</tr>
<tr>
<td>English (EN)</td>
<td>9844</td>
<td>0.33</td>
</tr>
<tr>
<td>Spanish (ES)</td>
<td>33164</td>
<td>0.51</td>
</tr>
<tr>
<td>Finnish (FI)</td>
<td>48644</td>
<td>0.22</td>
</tr>
<tr>
<td>French (FR)</td>
<td>85678</td>
<td>0.20</td>
</tr>
<tr>
<td>Dutch (NL)</td>
<td>45593</td>
<td>0.51</td>
</tr>
<tr>
<td>Polish (PL)</td>
<td>21436</td>
<td>0.70</td>
</tr>
<tr>
<td>Slovenian (SL)</td>
<td>6098</td>
<td>0.16</td>
</tr>
</tbody>
</table>

4 Experiments and Results

4.1 Experimental Settings

We experimented in four different settings, named ‘Word’, ‘Unigram’, ‘Bigram’ and ‘Trigram’. Each refers to a setting in which a particular feature described in Section 3.3 is used i.e. ‘Word’ represents the setting where complete words are used as the only feature, while ‘Unigram’, ‘Bigram’ and ‘Trigram’ to the setting where generated uni-, bi- and tri-gram counts are used as features respectively.

In each of the settings, we kept the same division of training and test datasets as in the ICDAR 2019 competition to make the results directly comparable.

As for the machine learning models, we used sklearn’s ‘DictVectorizer’, ‘OneHotEncoder’, and ‘CountVectorizer’ for data transformations and ‘SVC’ classifier with default parameters for training and testing. The only optimization was done by setting the class_weight to ‘balanced’ to overcome the issue of imbalanced class distribution in the training data for some of the languages (more detail about it in the following results section).

4.2 Results and Discussion

The results of the experiments are presented in Table 4 for all four settings i.e. ‘Word’, ‘Unigram’, ‘Bigram’, and ‘Trigram’. For comparisons, the results from the ICDAR2019 post-OCR error detection task are given in Table 3. The scores are highlighted in bold if they are better than the ICDAR2019 results for the corresponding language in each individual setting, while the best score across all four settings is underlined. If a score is both bold and underlined, it means it is the state of the art score.
As can be seen, with ‘Word’, our system beats the ICDAR2019 F1 scores for five languages (BG, EN, ES, NL, PL), while ICDAR2019 scores are better for three language (CZ, DE, FI, SL). Due to the time constrains we could not complete the experiments for the remaining two languages (FR, FR).

With ‘Unigram’, our system beats the ICDAR2019 F1 scores for six languages (BG, EN, ES, FR, NL, PL) with the default class distribution (more details about class distribution to follow). With the balanced class distribution, we were able to beat scores of two more languages (CZ, SL) making it eight in total. For the remaining two languages (DE, FI), ICDAR2019 results are better. Also not that we get noticeable improvements in F1 scores with ‘Unigram’ as compared to ‘Words’ for all of the languages except NL and PL.

With ‘Bigram’, our system beats the ICDAR2019 results for seven languages (BG, CZ, EN, ES, FR, NL, PL) with default class distribution. Again, due to time constraints, we were unable to complete the experiments with the balanced class weight in the ‘Bigram’ setting, but similarly to the other results in the ‘Unigram’ setting, we should expect an improvement once the experiments have been completed. The languages for which our approach does not beat the ICDAR2019 scores are Finnish and Danish. There is a marginal difference for Finnish (0.83 vs 0.84), while a notable difference for Danish (0.81 vs 0.95). Also note, we achieved further improvements compared to the ‘Unigram’ F1 scores.

With ‘Trigram’ the results start deteriorating for most of the languages, and improve for a couple of them (FR and NL) achieving state-of-the art for NL.

In summary, we were able to beat ICDAR2019 results for eight out of ten languages in different settings reaching state-of-the art F1 scores (underlined and bold) for those languages. The improvements vary from +0.2 to +2.1 for Bulgarian (0.77 to 0.86), Czech (0.70 to 0.78), English (0.67 to 0.83), Spanish (0.69 to 0.90), French (0.67 to 0.83), Dutch (0.71 to 0.87), Polish (0.82 to 0.84), and Slovenian (0.69 to 0.74).

As can be noticed in all four settings, we get
comparatively low F1 scores for CZ, DE, and SL. The reason for this is the low recall resulting from imbalanced class distribution in the training data of these languages. The class distribution for the class label 0/1 is 0.19/0.81, 0.29/0.71, 0.16/0.84 and 0.20/0.80 respectively for CZ, DE, SL, and FI as also shown in the Table 2. The sklearn’s SVC classifier provides an option to balance the class weight by setting its class_weight parameter to ‘balanced’. With this optimization the results improved for CZ, FI, SL as shown in Table 4 in the ‘Unigram’ settings while declined for the rest of the languages.

5 Conclusion and Future Work

Training supervised machine learning models with large number of features is a computationally expensive task. This has been demonstrated in previous work where handcrafted features were considered at the expense of high computational costs. In this study we have taken a different approach and have proposed to use n-gram counts as the only feature to train SVM models. N-gram counts have previously been used for post-OCR detection, but not in the sense that we have proposed in this study. Instead of computing the n-gram counts over the entire training data we have proposed to compute them within a given token and use them as the only feature to train and test our models.

To find the best ‘n’ for the n-grams, we experimented with uni-grams, bi-grams, and tri-grams. We tested the approach on the ICDAR 2019 dataset. As the experiment results show, with uni-grams our model outperforms the best system reported in the ICDAR2019 competition for six out of the 10 European languages included in the competition with default class distribution, and two more languages with balanced class distribution. With bi-grams, our model outperforms the previous system for 7 languages with comparable results for the remaining three languages. With tri-grams the results start deteriorating for most of the languages meaning that bi-grams are the best choice. Overall, with this approach we were able to beat 8 out of the 10 languages achieving state-of-the-art results for those languages.

The proposed approach is interesting because it eliminates the need for feature engineering; a task which is laborious and computationally expensive. The results show simple n-gram counts, which are fairly easy to compute, are enough for the task at hand. The approach is also gainful because it does not require large amounts of data. Given the relatively small datasets we experimented with we were able to show our method is performing better for the majority of languages compared to deep learning systems such as the ones explored by the CCC and UVA teams.

As previously said, we have used an SVM classifier with default parameters. In the future, we plan to apply parameter optimization, e.g. scaling, class distribution, grid-search, etc. to try and improve the results further. For example to detect real word errors when the word with an error is still a good word but inappropriate in the context. Another possible way to improve the results is to use the back-off approach in the n-gram setting. Taking a back-off approach we will use a bi-gram if a tri-gram is not in the vocabulary in a tri-gram setting, and likewise a uni-gram if a bi-gram is not in the vocabulary.

With these reported state-of-the-art results on post-OCR error detection, it will be interesting to experiment how these results will contribute to the improvement of the overall task of post-OCR error correction. We leave this to be another possible direction to explore further.

Acknowledgments

The work presented here was funded by (1) the Dictionary/Grammar Reading Machine: Computational Tools for Accessing the World’s Linguistic Heritage (DReAM) Project awarded 2018–2020 by the Joint Programming Initiative in Cultural Heritage and Global Change, Digital Heritage and Riksantikvarieämbetet, Sweden; (2) the Swedish Research Council as a part of the project South Asia as a linguistic area? Exploring big-data methods in areal and genetic linguistics (2015–2019, dnr 421-2014-969); (3) From Dust to Dawn: Multilingual Grammar Extraction from Grammars project funded by Stiftelsen Marcus och Amalia Wallenberg Minnesfond 2007.0105, Uppsala University; (4) the Swedish Research Council as part of the project Evaluation and refinement of an enhanced OCR-process for mass digitisation (2019–2020, dnr IN18-0940:1 and 421-2014-969). It is also supported by Sprkbanken Text and Swe-Clarin, a Swedish consortium in Common Language Resources and Technology Infrastructure (CLARIN) Swedish CLARIN (dnr 821-2013-2003).
References


A Data-Driven Semi-Automatic Framenet Development Methodology

Shafqat Mumtaz Virk
Språkbanken Text, Dept. of Swedish
University of Gothenburg
405 30 Gothenburg, Sweden
virk.shafqat@gmail.com

Dana Dannélls
Språkbanken Text, Dept. of Swedish
University of Gothenburg
405 30 Gothenburg, Sweden
dana.dannells@svenska.gu.se

Lars Borin
Språkbanken Text, Dept. of Swedish
University of Gothenburg
405 30 Gothenburg, Sweden
lars.borin@svenska.gu.se

Markus Forsberg
Språkbanken Text, Dept. of Swedish
University of Gothenburg
405 30 Gothenburg, Sweden
markus.forsberg@svenska.gu.se

Abstract

FrameNet is a lexical semantic resource based on the linguistic theory of frame semantics. A number of framenet development strategies have been reported previously and all of them involve exploration of corpora and a fair amount of manual work. Despite previous efforts, there does not exist a well-thought-out automatic/semi-automatic methodology for frame construction. In this paper we propose a data-driven methodology for identification and semi-automatic construction of frames. As a proof of concept, we report on our initial attempts to build a wider-scale framenet for the legal domain (LawFN) using the proposed methodology. The constructed frames are stored in a lexical database and together with the annotated example sentences they have been made available through a web interface.

1 Introduction

Frame semantics is a theory of meaning in natural languages proposed by Charles Fillmore (Fillmore, 1977, 1982). The theory stipulates that meanings of words can be best understood with reference to the situations they invoke in the minds of the speakers. The concrete manifestation of frame semantics is the computational lexical resource FrameNet, first constructed within the English Berkeley FrameNet (BNF) project (Baker et al., 1998). The resource has inspired work on framenets for many other languages and domains, including Chinese (You et al., 2007), French (Candito et al., 2014), German (Burzachetti et al., 2009), Hebrew (Hayoun and Elhadad, 2016), Korean (Kim et al., 2016), Italian (Lenci et al., 2010), Japanese (Saito et al., 2008), Portuguese (Toro, 2013), Spanish (Díaz, 2009) and Swedish (Borin et al., 2010a; Dannélls et al., Forthc.a).

FrameNet is built around the notion of frames. A frame in this context is a schematic representation of events, objects, situations, institutions, etc. The primary components of a frame are the frame definition along with the frame elements and the lexical units documented with corpus evidence (Ruppenhofer et al., 2016).

There are at least three major framenet development strategies that have been reported in the literature (Candito et al., 2014) namely: (1) lexicographic frame-by-frame (2) corpus driven lemma-by-lemma (3) the full-text strategy. Of these three, the most commonly applied strategy is frame-by-frame, where the frame is defined along with its frame-elements first. Example sentences are then chosen from a corpus for a disambiguated annotation of lexical units and frame-elements. In the lemma-by-lemma strategy, a set of lemmas (lexical units) are chosen first and then all their occurrences are annotated in a given corpus together with the annotation of frame-elements. In the full-text strategy, all content words in a given text are annotated. The latter two strategies pre-suppose that frames already exist, although new frames can also be developed as new senses of lexical units are encountered in the text.

All three strategies involve exploration of corpora and a fair amount of manual work for construction of semantic frames, and for searching suitable example sentences to be annotated. Various framenet development projects have reported the use of (or developed) tools to assist in the corpus exploration and frames construction parts (e.g. the Swedish FrameNet++ project; Borin et al., 2010b;
lines for frame construction (Burchardt et al., 2009; Ruppenhofer et al., 2016). However, there is no well-thought-out automatic/semi-automatic procedure for frame development from scratch.

In this study, we propose a data-driven method for the identification and semi-automatic construction of domain-specific semantic frames. The identification involves recognizing the domain-specific events, entities, relations, procedures, etc. for which the semantic frames can be constructed. The development part involves discovering various semantic roles of a given frame, spotting and linking lexical units to the target frame, and annotating example sentences.

2 Data and Pre-processing

As a proof of concept of our methodology, we prepared a small data-set by collecting a set of documents from the legal domain and applied OCR processing to produce a machine readable version. The data was then enriched with metadata and linguistic annotations. It was then stored and made accessible through a corpus infrastructure tool for easy excess and exploration.

The data used in this study was downloaded from web repository of the united nations high commissioner for refugees. A subset of 900 pdf documents resulting from the hits for the search string “well-founded fear of being persecuted” were downloaded and then OCRed using ABBYY FineReader software. Each document was processed for the following document, token, and structure level attributes.

- **Text-Level Attributes:** Article title, author, publisher, topic, and country.
- **Token-Level Attributes:** word tokenization, lemmatization and part of speech (POS) tags.
- **Structure-Level:** Sentence segmentation and dependency parses.

Document level attributes were preserved while downloading the documents, and for various token and structure level annotations we used Sparv (Borin et al., 2016), which is an annotation pipeline developed and maintained at Språkbanken Text.

It can be used to automatically annotate textual data with various token, text, and structure level attributes using in-house and third party annotation tools. We used Stanford’s NLP toolkit (Manning et al., 2014) for token and structure level annotations. After annotating the data, it was made available through Korp (Borin et al., 2012), a state-of-the-art corpus infrastructure tool developed and maintained by Språkbanken Text. It provides various basic and advanced level features to better search, explore, and visualize a corpus. Figure 1 shows a screenshot of the search results together with various annotations when searched for a simple string ‘article’. The left-hand pane shows the sentences containing the search string, and the right-hand pane shows various text level (title, author, topic, etc.) and token level (lemma, pos-tag, etc.) tags for the selected token (the word ‘title’ highlighted with black background). The bottom right corner also shows the dependency parse tree of the sentence.

The extended search tab provides options to search for any of the individual attributes (e.g. author, title, pos-tag etc.) or a combination of them combined by and/or logical operators. The advanced tab allows the user to formulate a search query using the CQP query language (Christ, 1994). The data is password protected due to IPR issues and is intended to be used for internal research purposes at this stage. However, in the future we have plans to release it with suitable licensing options.

3 Frame Development Methodology

A step-wise description of the frame development methodology is given below. This is followed by construction of an example frame using the described methodology. The developed frame is a part of the legal domain framenet (more details about the legal domain framenet in Section 4).

1. As a first step, the word segmented data reported in Section 2 is sorted in the descending order of term frequencies after removing the function words. The purpose is to identify the

---

1[www.refworld.org](http://www.refworld.org)

2This search string was chosen in connection to another part of the project with its own objectives of analyzing the ill-treatment of refugees.

3[https://www.abbyy.com/](https://www.abbyy.com/)

4Språkbanken Text is a research unit and also forms part of Nationella språkbanken (the National Language Bank), a national e-infrastructure supporting research based on language data in Sweden.

5Here, we use “term” in a basically non-technical sense, to
most frequently used terms and cover them first while designing the frames.

2. Each item in the obtained list is inspected manually to decide if it is a domain-specific term or not with the help of a domain expert if required. In a better setting, the word list can be filtered out beforehand with the help of any already existing domain-specific terminology list or a lexical database such as WordNet (Fellbaum, 1998) to speed up the process and reduce the manual work, but we leave this aspect to be explored in the future.

3. Once a term has been determined to be a domain-specific term, the next step is to check if the term fits as a lexical unit of an already existing frame or not. If it does, it is simply added as a lexical unit of the existing frame. If not, it means a new frame needs to be designed. A suitable name for the frame is chosen, and the following procedure is followed to identify various frame-elements of the frame. Let the term be T.

(a) All sentences containing the term T are extracted from the data and their dependency parses are retrieved from the parsed data reported in Section 2.

(b) From the dependency parses, the head and dependent text segments are grouped together for each of the dependency relations.

(c) The text segments in each group are then manually inspected and a decision is made to whether we need a frame element or not.6 If yes, a suitable frame-element title is chosen, and the element is made part of the frame. This step is repeated for each dependency relation.

Let’s now walk through the construction of an example frame Article using the above described methodology. In our data collection, the term ‘article’ occurred 29 times, and it is easy to recognize that this term has a legal domain sense in addition to four others as per WordNet. Considering that no frames have been designed previously, we need to construct a frame for which the word ‘article’ will be a lexical unit. This is step 2 of our methodology. We chose the name Article for the frame, and move towards construction of its structure i.e. step 3. Table 1 shows text segments generated using step 3 for the syntactic relations ‘dobj’ (i.e. direct object)7 and ‘nmod’ (nominal modifier). For space reasons, only a few entries for a selected set of relations are shown.

A manual inspection of the text segments under the ‘Dependent’ for the ‘dobj’ relation reveals that the word ‘article’ whenever used in the legal domain is often followed

---

6This is basically a linguistic decision which requires both experience of framenet development and training in grammatical and lexical-semantic analysis for the language in question ([legal] English in our case). See Ruppenhofer et al. (2016) for a more detailed discussion of the considerations and reasoning involved.

7The constituents in question are not direct objects; rather, the relation should be labeled ‘flat’ or possibly ‘appos’ under the UD scheme. This does not matter to our example, but it will of course become important when we will be looking to
by a figure or figures that refers to the article number of the constitution/agreement. This means we need a frame-element to capture and represent this usage. It can perhaps be argued that it is so obvious that the term ‘article’ will be followed by an article number so why we need to follow the described procedure. It may appear obvious in this case, but it can not be generalized across frames and frame-elements (imagine the construction of the frame *Injunction* which refers to a judicial order restraining a person from beginning or continuing an action). Our experience suggests that this usage based methodology is very helpful in identifying the frames and frame-elements.

Similarly, a manual inspection of the text segments under the ‘Dependent’ for ‘nmod’ relation revealed that there is often a mention of a law/protocol/constitution to which the article belongs e.g ‘of the Turkish constitution’. This means we need a frame-element to capture that information, hence, the frame-element *Constitution*. The same procedure was followed to design the frame-elements *Interpretation*, and *Date* while inspecting the ‘acl:relcl’ (i.e. relative clause modifier) and ‘nmod:tmob’ (i.e. temporal modifier) relations respectively.

For the actual frame construction and storage, we used *Karp*, which is another lexical infrastructure tool developed and maintained at Språkbanken. Figure 2 shows a screenshot of the tool and the structure of the *Article* frame. It also shows a set of annotated examples and the lexical units which can trigger this frame.

4 A Legal Domain FrameNet and its Applications

4.1 LawFN

General-language lexical resources such as BFN are both too broad and too narrow for successful deployment in domain-specific natural language processing (NLP) applications. On the one hand, they contain more than one sense for many headwords, most of which are not relevant in the domain of interest (but which lower the accuracy of the analysis). On the other, they often lack some important domain-specific usages. For this reason, several domain-specific frame-nets have been compiled, covering e.g. medicine, football/soccer and tourism (Borin et al., 2007; Schmidt, 2009; Torrent et al., 2014).

There have been some initiatives reported in literature on building frame-nets for the legal domain (e.g. Venturi, 2011; Bertoldi and Chishman, 2012). However, to the best of our knowledge, these efforts have been quite limited in scope, and no full-scale resource of the kind proposed here has been presented. In this study, we report on our initial attempts to build a wider-scale FrameNet for the legal domain that we call LawFN.

4.2 Applications of LawFN

The major motivation behind initiating the development of LawFN is its potential applications in the area of Legal Tech, which refers to the use of technology and software for providing various legal services and support to the legal industry. In recent years, Legal Tech has gained popularity, and the use of technology is increasing in many legal-domain activities such as case management, contract management, document automation and analysis (Gruzauskas and Ragavan, 2020), etc. Some of these tasks require semantic analysis of the text within legal domain documents.

When we apply currently available semantic analysis technology to extract information about laws automatically from text we do not retrieve the desired analysis. For example when we analyze the sentence: “Justice Kirby similarly stated in the same judgement that the convention does not require or imply the elimination by the state of all risks of harm; rather it posits a reasonable level of protection, not a perfect one.”, with the general framework for semantic role labeling (Punyakanok et al., 2008), we get the analysis as shown in Example 1.

(1)  

As Example 1 shows, the identified LU is ‘state’, and the frame elements are *Announcer*, *Discourse* [stated] *Location* [that the convention does not require or imply the elimination by the state of all risks of harm; rather it posits a reasonable level of protection, not a perfect one.] *Utterance*.
### Table 1: Frame Elements of the Article Frame

<table>
<thead>
<tr>
<th>Head</th>
<th>Dependent</th>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>'articles'</td>
<td>'20 and 30'</td>
<td>'The recast Directive continues.... “mental health problems” and the Parliament’s “mental health illnesses” under articles 20 and 30.'</td>
</tr>
<tr>
<td>'articles'</td>
<td>'6, 7 and 8 of the Statute'</td>
<td>Particularly relevant for exclusion are articles 6, 7 and 8 of the Statute, ....'</td>
</tr>
<tr>
<td>'articles'</td>
<td>'2 to 34 inclusive of the Convention'</td>
<td>'The States Parties to the present Protocol undertake to apply articles 2 to 34 inclusive of the ....'</td>
</tr>
<tr>
<td>'articles'</td>
<td>of the Convention</td>
<td>With respect to those articles of the Convention to be applied ..Federal States:'</td>
</tr>
<tr>
<td>'article'</td>
<td>of the European Convention on Human Rights</td>
<td>'They have been given leave to enter the United Kingdom because article 3 of the European Convention on Human Rights forbids their return ... punishment there.'</td>
</tr>
<tr>
<td>'article'</td>
<td>of the Turkish Constitution</td>
<td>The essential point..... under article 24 of the Turkish Constitution.'</td>
</tr>
<tr>
<td>'article'</td>
<td>which embodies an immediate obligation to respect and ensure all of the relevant rights</td>
<td>In this sense the obligation differs significantly from that contained in article 2 of the International Covenant on Civil and Political Rights which embodies an immediate obligation to respect and ensure all of the relevant rights.'</td>
</tr>
<tr>
<td>'article'</td>
<td>which guarantees &quot;women equality with men before the law&quot;</td>
<td>This contravenes the right of women under . article 15(9) of the Convention on the Elimination of Discrimination against Women, which guarantees &quot;women equality with men before the law.'</td>
</tr>
</tbody>
</table>

## 4.3 Frame Types in LawFN

For a better organization, we divide frames in the LawFN into two types: Entity frames and Event frames. Entity frames are simpler in their structure and they are meant to represent various legal domain entities e.g. judge, court, tribunal, etc. Event frames are a bit more complex in their structure and are meant to represent various legal domain events or processes such as prosecution, judgement, defense, etc. In another sense, entity frames act as slot fillers for various semantic roles of an event frame. Consider Figure 3 showing an annotated sentence in which a filler frame (i.e. Judge) files in a semantic role of an event frame (i.e. Judgement).

In the annotated sentence, the frame Judge is triggered by the lexical unit ‘justice’ and NAME is the only frame element which is realized in this sentence (the annotation in red color). This frame then becomes frame-element JUDGE for the semantic frame Judgement triggered by the lexical unit ‘judgement’. The Judgement frame is the only other realized frame-element (annotation in green).

### 4.4 Current Status of LawFN

Using the proposed frame-development methodology, we have developed a total of 10 frames, containing 36 frame-elements, 24 lexical units, and 22 annotated example sentences. A list of constructed frames together with their frame-elements, lexical units, and annotated example sentences are provided in Appendix A.

## 5 Conclusions and Future Work

Our contribution is twofold. First, we have reported a semi-automatic frame-development methodology, which can help speed up the frame development process. Second, we have reported initial attempts in building a framenet for the legal domain and
In the future, we have plans to continue constructing more frames. Our dataset is quite limited in scope and size, which we plan to extend in the near future. With the bigger data set, we will be able to explore more syntactic patterns. This means capturing more possible use cases, and hence being in a position to better design the frames. Also, we have plans to annotate data with the developed frames, and build a frame-semantic parser exploiting machine learning. The developed parser then can be used for automatic semantic analysis of legal domain documents, which then can be used to extract particular types of information from those documents. As an example, suppose one is interested to extract all judgements made by a particular judge under a particular law/protocol/agreement from a corpus of court decisions. Parsing the corpus for Judgement, Law, and Article frames can help extract that information automatically. Evaluation of the extracted information for accuracy is another direction that we plan to explore in the future.

Acknowledgments

The work described here has been funded in part by the Swedish Research Council and by the University of Gothenburg, through their support of the Swedish national research infrastructure The National Swedish Language Bank (SRC grant 2017–00626) and of Språkbanken Text, and in part by From Dust to Dawn: Multilingual Grammar Extraction from Grammars project funded by Stiftelsen Marcus och Amalia Wallenbergs Minnesfond 2007.0105, Uppsala University.

References


Giulia Venturi. 2011. Semantic annotation of
italian legal texts: A framenet-based ap-
proach. Constructions and Frames 3:46–79.
https://doi.org/10.1075/cf.3.1.02ven.

Liping You, Tao Liu, and Kaiying Liu. 2007. Chi-
nese Framenet and OWL representation. In
Sixth International Conference on Advanced
Language Processing and Web Information
Technology (ALPIT 2007), pages 140–145.
https://doi.org/10.1109/ALPIT.2007.50.
### A Developed Frames

<table>
<thead>
<tr>
<th>Frame</th>
<th>Frame-Elements</th>
<th>Lexical Units</th>
<th>Annotated Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Article</td>
<td>Number, Interpretation, Constitution, Date</td>
<td>article.n</td>
<td>The States Parties to the present Protocol undertake to apply [articles]LU 2 to 34 inclusive of the [Convention to refugees]Constitution as hereinafter defined.</td>
</tr>
<tr>
<td>Act</td>
<td>Act, Purpose, Consequence, Type</td>
<td>act.n, act.v, deed.n, action.n</td>
<td>Such violence must be given a broad interpretation and may be defined as any [act]LU of [gender-based violence]Act that results in, or is likely to result in, [physical, sexual or psychological harm or suffering to women]Consequence, including threats of such acts, Confer/UNHCR003.</td>
</tr>
<tr>
<td>Injunction</td>
<td>Type, Injunction, Court, Date</td>
<td>injunction.n</td>
<td>In [July 2008]Date the <em>ECHR</em>, acting on a complaint filed by the BHC, issued an [interim]Type [injunction]LU [halt the planned demolition of Romani housing in Sofia]Injunction.</td>
</tr>
<tr>
<td>Judge</td>
<td>Name, Designation, Court</td>
<td>judge.n, justice.n</td>
<td>Justice LU [Kirby]Name similarly stated in the same judgement that the Convention does not require or imply the elimination by the State of all risks of harm; rather it &quot;posits a reasonable level of protection, not a perfect one&quot;.</td>
</tr>
<tr>
<td>Judgement</td>
<td>Judgement, Type, Judge</td>
<td>judgement.n</td>
<td>Justice Kirby similarly stated in the same [judgement]LU that [the Convention does not require or imply the elimination by the State of all risks of harm; rather it &quot;posits a reasonable level of protection, not a perfect one&quot;]Judgement.</td>
</tr>
<tr>
<td>Defense</td>
<td>Victim, Defender, Law, Type, Place, Date, Charges</td>
<td>defence.n, defend.v</td>
<td>It is reasonable to conclude, therefore, that the purpose of enacting section 31 was to meet the difficulties exposed by the judgments in Adimi by incorporating into domestic law, with certain modifications, the principles contained in article 31 in the form of a [defence]LU to [the charges]Charges most likely to be brought against [asylum seekers]Victim entering the country on false passports.</td>
</tr>
</tbody>
</table>
A Deep Learning System for Automatic Extraction of Typological Linguistic Information from Descriptive Grammars

Shafqat Mumtaz Virk
Språkbanken Text, Dept. of Swedish
University of Gothenburg
405 30 Gothenburg, Sweden
virk.shafqat@gmail.com

Daniel Foster
University of Gothenburg
405 30 Gothenburg, Sweden
daniel.a.foster@gmail.com

Azam Sheikh Muhammad
Chalmers University of Technology
412 96 Gothenburg, Sweden
sheikhazam@gmail.com

Raheela Saleem
Gift University
Gujranwala, Pakistan
raheela.saleem50@yahoo.com

Abstract

Linguistic typology is an area of linguistics concerned with analysis of and comparison between natural languages of the world based on their certain linguistic features. For that purpose, historically, the area has relied on manual extraction of linguistic feature values from textual descriptions of languages. This makes it a laborious and time expensive task and is also bound by human brain capacity. In this study, we present a deep learning system for the task of automatic extraction of linguistic features from textual descriptions of natural languages. First, textual descriptions are manually annotated with special structures called semantic frames. Those annotations are learned by a recurrent neural network, which is then used to annotate un-annotated text. Finally, the annotations are converted to linguistic feature values using a separate rule based module. Word embeddings, learned from general purpose text, are used as a major source of knowledge by the recurrent neural network. We compare the proposed deep learning system to a previously reported machine learning based system for the same task, and the deep learning system wins in terms of F1 scores with a fair margin. Such a system is expected to be a useful contribution for the automatic curation of typological databases, which otherwise are manually developed.

1 Introduction and Background

Linguistic typology is an area of linguistics concerned with analysis of and comparison between natural languages of the world in terms of their structural and functional attributes. Among others, major objectives of the area are exploring the range of possibilities for expressing different linguistic categories, trying to understand the extent to which the presence of different features depend on one another in larger patterns, and to study how the global distribution of language traits has come about through an interplay of tendencies inherent to languages and historical contingencies (Bickel, 2015). For achieving such aims, historically, the area has relied heavily on scholars having to read textual documents (commonly known as descriptive grammars) describing languages, manually extracting values of a pre-defined set of features, and then comparing languages based on the extracted feature values. To make the whole exercise systematic and structured, traditionally, the features are expressed in the form of questions e.g. ‘What is the order of adjectives and noun in language X?’, and scholar’s job is then to find answers to such questions by reading descriptive documents about language X. The answers are often formulated to be simple strings e.g. ‘NA’, ’AN’, or ’Both’ representing the fact that nouns precede adjectives, nouns follow adjectives, or ’both nouns may follow or precede adjectives’ respectively in the case of above give question. For easy storage and retrieval, the questions and their answers are recorded in special kind of databases known as typological databases. There exist many such databases and a fuller list is available at languagegoldmine.com/. These databases are later used to compare and analyze languages to achieve the above mentioned objectives of the area. In addition, information in such databases has proven to be useful for a number of natural language processing (NLP) related tasks. A survey on the usefulness of typological information in NLP can be found in (O’Horan et al., 2016).

As can be imagined, the manual development of typological databases is an expensive enterprise both in terms of time and efforts, and also their qual-

Proceedings of Recent Advances in Natural Language Processing, pages 1484–1493
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_167
ity and coverage is bound by human brain capacities. Extensive digitization efforts (e.g., (Michel et al., 2011)) and the advancement of computational methodologies, including NLP, offer many possibilities for easing the task of developing linguistic typological databases. A maximally automated approach would allow for the generation of such databases at a hitherto unparalleled scale, increasing both the number of features and languages that can be analyzed and compared taking the area to new heights. However, this requires developing methodologies and systems for automatic extraction of typological linguistic information from descriptive documents. This exactly is a major objective of the study reported in this paper.

Previously, a few approaches and systems have been reported for the automatic extraction of typological information, but a practical system still remains to be be developed. Pattern matching and machine learning based classification are the two main computational paradigms that have been exploited so far (see Section 3). Pattern matching has its own limitations as it is impossible to design patterns which can cover every possible scenario. Similarly, machine learning classification algorithms are heavily dependent on feature engineering, which have its own limitations and features are often expensive to compute. To address some of these issues, we report a deep learning based system in this study (Section 4). First, textual descriptions are manually annotated (Section 4.1) with special structures called semantic frames, which are based on the theory of frame-semantics (Section 2). Those annotations are learned by a recurrent neural network using word embeddings learned from general purpose text as the major source of knowledge (Section 4.2). The trained model is used to annotate the un-annotated data, and the annotations are then converted to linguistic feature values using a separate rule based module (Section 8). Our frame annotation system beats previously reported systems on a test set with a fair margin in terms of F1-scores.

2 Frame-Semantics, FrameNet, and Frame-Semantic Parsing

Frame semantics is a theory of meanings in natural languages (Fillmore, 1982). It stipulates that meanings of words can be best understood with reference to the situations they invoke in the minds of the speakers. The concrete manifestation of frame semantics is a computational lexical resource called a framenet. The first such resource was the English Berkeley FrameNet (BNF) (Baker et al., 1998) which has inspired work on framenets for many other languages. The “lexical entry” in a framenet is called a (semantic) frame, which is a script-like description of a prototypical event, object, relation, or scenario. A frame consists of triggers – the lexical units (words) which evoke a specific situation – and additional components called frame-elements that fills in various semantic slots of the frame. Below is an example sentence manually labeled with the COMMERCE_SELLING semantic frame and its frame-elements i.e. seller, buyer, and price. The frame is triggered by the word ‘sold’.


The process of automatically performing the above type of annotation/analysis is called semantic parsing. The first such frame semantic parser was proposed by Gildea and Jurafsky (2002), which has been followed by a number of other systems/approaches (e.g., (Das et al., 2014; Roth and Lapata, 2016)).

3 Related Work

Previously, a few experimental techniques and associated systems have been reported for automatic extraction of typological information. In (Borin et al., 2018; Virk et al., 2017), the authors have reported on simple pattern matching and syntactic parsing based systems. The systems have modest accuracy and recall and are very restricted with respect to the number of features they can target.

To overcome some of those limitations, in (Virk et al., 2019) the authors exploited frame-semantics and machine learning approaches to build a system that can extract information about a few experimental features. The systems is based on the work reported in (Malm et al., 2018) where the authors proposed to use frame-semantics to represent the typological linguistic information. They also developed a domain specific framenet (LingFN) containing semantic frames for the linguistic domain representing various linguistic terms and phenomena (e.g. verb, noun, agreement, inflection, etc.). The developed LingFN was used by Virk et al. (2019) to annotate textual descriptions of languages with linguistic domain frames (more details in Section 4.1) and develop an automatic typological information extraction system using machine learning.
In (Søren and Rama, 2019), the authors reported a two step strategy to detect the parts of the text that possibly contains value of a given feature, and then extracting the feature value from it. For the first step their approach relies on keyword spotting and pattern matching, while on machine learning classification for the second step.

In (Hammarström, 2020), the author reported a simple keyword based approach for extracting values of one typological feature about tones for many languages from thousands of documents with an overall accuracy of 89.1%.

4 Proposed System

Figure 1 shows the complete architecture of the system that we propose. As shown, it has a clear division between three components: data annotation, deep learning, and typological information formulation. In the following subsection, the first two components will be explained, while the explanation of the third component is deliberately deferred until Section 8 for a better flow.

4.1 Data

A small corpus consisting of descriptive grammars of the natural languages spoken in South Asia was reported in (Borin et al., 2018), and a set of documents from that corpus annotated with LingFN frames was reported in (Virk et al., 2019). Annotation of a descriptive grammars with LingFN frames involve identification of verbal units and selection of appropriate linguistic semantic frames and their frame elements. As a part of the study reported in this paper, the annotated corpus was extended resulting in a total of 70 annotated documents (a document corresponds to 3 to 7 pages of text). Figure 2 shows an annotated sentence. As can be seen, the sentence is annotated with two linguistic domain frames i.e. VERB triggered by the word ‘verb’ having ‘data’ and ‘data_translation’ frame elements and the frame ‘AFFIXATION’ triggered by the word ‘suffixed’ and having ‘degree’ and ‘anthromorphic_entity’ frame elements. We refer the reader to (Virk et al., 2019) for more details on annotations and the annotation process. The data used in this study consists of a total of 70 documents comprising around 3,986 sentences, 7,170 semantic frames, and 4,669 frame-elements.

4.2 Deep Learning Part

In the system developed here, manually labeled data (reported in previous subsection) is used to train a deep learning model which is then used to label un-annotated data. The architecture of model is similar to the one proposed by Swayamdipta et al. (2017) using the RNNs. Figure 3 shows a simplified version of the architecture together with various inputs (features). To make the description self explanatory, we briefly explain here both the inputs and the architecture with respect to an example input sentence: ‘Nouns agree with the adjectives’. The word ‘agree’ triggers the frame AGREEMENT (shown in red), while ‘Nouns’ and ‘the adjectives’ represent two text segments (in purple). These text segments fill in the roles of two frame-elements, i.e. ‘Segment_1’ and ‘Segment_2’, to be learned and later predicted by the deep learning model. The model uses two bilingual LSTM networks (biLSTM) and one LSTM for various computations as described below.

- Each word at position $q$ in the input sentence is converted to a vector:

$$v_q = [d_q; e_q; o_q; γ_q]$$

(1)

where $d_q$ is the learned embedding\(^1\) of the word type, $e_q$ is a pre-trained embedding of the word type, $o_q$ is the learned embedding of the part-of-speech tag of the word, and $γ_q$ is the distance of the word from the beginning of the target (the word triggering a frame).

- These word representations are given as input to a bidirectional LSTM (biLSTM), each of whose hidden state then becomes a contextualized representation of the following form.

$$h_q^{ok} = [biLSTM^{tok}(v_1, v_2, ..., v_n)]$$

(2)

- These token representations are then used to compute contextualized representation of various spans of the sentence as shown below:

$$h_{(i,j)}^{span} = [biLSTM^{span}(h_1^{ok}, v_2^{ok}, ..., h_j^{ok})]$$

(3)

\(^1\)In this study, we use Stanford’s GloVe (Global Vectors for Word Representation) word embeddings (Pennington et al., 2014) as proposed by Swayamdipta et al. (2017). GloVe embeddings were created from data from Wikipedia and newswires.
The token hidden states given above (Eq 2) together with an LSTM are used to compute a contextualized target representation as shown below:

$$v_t = \left[ \text{LSTM}_{\text{target}}(h_{t_{\text{start}}-1}, \ldots, h_{t_{\text{start}}+1}) \right]$$  \hspace{1cm} (4)

The target representation, together with the learned frame embedding $$v_f$$ and the lexical unit embedding $$v_l$$, are used to represent segments as:

$$v_{f,l,t} = [v_f; v_l; v_t]$$  \hspace{1cm} (5)

And, for every segment, a segment score is computed as:

$$v_s = [h_{i,j}^{\text{span}}; v_l; \mu]$$  \hspace{1cm} (6)

Where $$v_s$$ is a learned embedding of a segment at position $$(i, j)$$ and $$\mu$$ represents two other features: the length of the span, and the span’s position with respect to the target.

This is then passed through a rectified linear unit to get a segment score as:

$$\phi(s, x) = w_2 \cdot \text{ReLU} \{ w_1 [v_s; v_{f,l,t}] \}$$  \hspace{1cm} (7)

The segment score then becomes part of a criterion, which the model is trained to maximize on the training data. Once trained, the model is used to predict labels for various spans of the input sentence. The experiments to label sentences using the learned models are reported in the next section. For more technical details on the deep learning model and its input, we refer the reader to (Swayamdipta et al., 2017).

5 Experiments

The data was divided into two major sets labeled as ‘Full’ and ‘Filtered’. The former is the data set where the annotation of all frames and frame-elements were preserved, while in the later the annotation of two problematic frame elements (i.e. ‘data’ and ‘data_translation’\(^2\)) were removed from the training data. The reason for removing

\(^2\)In LingFN, many frames have data and data_translation frame-elements, which are used to represent examples of various morphology or grammatical linguistic categories of the language described in the document. As an example consider the following annotation in which both data and data_translation frame-elements have been used to annotate an example numeral (i.e. ghri) and its translation (i.e. ‘one’): The numeral \([ghri]\_data, \text{[one]}\_data_translation\) is used as an indefinite article.
these two frame-elements is that they contain non-English words for which the embeddings are missing in the learned embedding space consequently impacting the model’s performance adversely. To show their impact, we conducted experiments with and without those frame-element annotations. With those two data sets, the experiments were conducted with the following three settings:

- **Word Embeddings**
- **Character Embeddings**
- **Mimic Embeddings**

In the first setting, word-embeddings were used to compute the token level representations (i.e. Eq 1), and consequently all other computations involving token representations. In the second setting, word-embeddings were replaced with character embedding, while in the third setting they were replaced with mimic embeddings. The motivation behind these three settings follows.

Since the word embedding space was learned from a general-purpose text, it understandably does not contain many of the the domain specific words in the data used in this study (i.e. the descriptive grammars). In addition, there are the non-English words, i.e the transliterated forms of the words of a specific language given as examples in the descriptive documents. In the annotated data, such words often appear as a part of the ‘data’ frame-element. For all those cases, a default word embedding was used by the system while learning and predicting the frame annotations. This means that the knowledge that words can bring to the system while learning was not available to the system resulting in a decreased system performance (as obvious from the results discussed in Section 6). As a solution to this out-of-vocabulary words issue, we experimented with the other two settings.

In the second setting, word embeddings were replaced with character embeddings while computing Eq 1, resulting in different token and span level representations (Eq 2 and 3). This was based on the intuition that even if a complete word has not been seen in the training set, the characters it contains have, hence, utilizing the character-level knowledge. This is particularly pertinent in the case of ‘data’ frame element. As is apparent from the results in given in Section 6, this technique improved the result considerably. A primary computational concern with this solution from the outset was the need to train character embeddings, which is computationally a highly expensive task. For that we relied on automatic conversion of a set of pre-trained word embeddings into character embeddings\(^3\). In this technique, for each word in a set of word embeddings, each character is given a vector inferred from the parent word. Then, for each word in the embeddings, when the character is seen again this vector is adjusted to reflect its average in the entire set. This technique is a useful workaround to get a meaningful set of character embeddings.

---

\(^3\)https://minimaxir.com/2017/04/char-embeddings/
In the third setting, word embeddings of the unknown words were inferred from the existing word-embedding space and used while computing input (Eq 1). The embedding inferring technique has been proposed by Pinter et al. (2017) in which a character-based bidirectional LSTM was used to infer vectors for a list of unknown words and to add them to the existing word-embeddings. They call their method the ‘mimick’, and it predicts an embedding for a word that should fit into the same space based on the sequence of character embeddings for the characters in the given word.

Originally, the authors used their technique in the context of OCR error correction and spelling variations issue, but in our case we have used it for the out-of-vocabulary issue. It appeared to be equally useful in our case as is obvious from the results given in the next section.

6 Results

Table 1 shows the results of three experimental settings discussed in the previous section. To show the impact of the ‘data’ and ‘data_translation’ frame-elements, results have been shown with and without them (i.e. ‘Full’ and ‘Filtered’ in the table). As can be seen, the system achieved a considerably low F-score when tested on the full data, which improved when the data and data_translation frame-elements were removed. The word to character level embedding replacement obtained an improvement from 46.9 to 51.3 on the full data, while from 55.0 to 58.8 without data and data_translation frames. This suggests that the character-level embeddings are a better choice in a domain specific setting even if there are no non-English transliterated words. As for the mimic technique, it deteriorates from 46.9 to 46.3 when applied on the full data, but improves from 55.0 to 57.0 when tested without the problematic data and data_translation frame elements. This suggests:

- The technique of inferring word embeddings and using them for the out-of-vocabulary issue is equally useful as it was for the OCR error and spelling variation issue.
- It is not advantageous to infer and use the embeddings of totally unrelated words i.e. the non-English transliterated words, such as words in the data frame element.

In summary, we achieved the best results with character-level embedding when applied to the full data, and with the mimic technique when data and data_translation were excluded.

7 Comparison to a Previous System

Table 2 shows comparison between the deep learning based vs a machine learning based system reported in (Virk et al., 2019) for the frame annotation task. The comparison is done on a separate smaller data set as the (Virk et al., 2019) system could not be run on the full data set due to memory issues arising from data size. As can be seen, both on the full and filtered data set versions, the proposed system beats the older system with a fair margin in term of F-scores. This proves the worth of such a system for domain specific frame semantic parsing which is to be used for typological information extraction as described in the next section.

8 Typological Information Formulation Module

In the proposed system, once a sentence has been automatically annotated with semantic frames and their frame-elements (the output from the second part of the system architecture), the annotations can be converted to a typological feature value as an answer to a typological question. Currently, we rely on a rule-based module for the conversion from annotations to feature values as explained in (Virk et al., 2019). This involves writing small modules as shown in Algorithm 1 given in Appendix A for converting annotations to feature values.

As can be seen the algorithm simply loops through the set of frames (line 2) and frame elements, and depending on their contents (line 3, 7, 8, 10, and 12), it assigns appropriate features value for the adjective-noun-order feature (line 9, 11, and 13) discussed in the introduction section. Similar modules can be used for other typological features.

Appendix B shows a set of order related feature and their values automatically extracted from descriptive grammar of ‘Ulwa’ language (Barlow, 2018) using the developed semantic parser, and the rule-based feature formulation module given in Appendix A. Note all features were extracted using only SEQUENCE semantic frame from LingFN.

4A semantic frame in LingFN which encodes ordering related information similar to the example given in the introduction section.
<table>
<thead>
<tr>
<th>Setting</th>
<th>Data</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word-Embeddings (baseline)</td>
<td>Full</td>
<td>0.52459</td>
<td>0.42440</td>
<td>0.46921</td>
</tr>
<tr>
<td>Word-Embeddings (baseline)</td>
<td>Filtered</td>
<td>0.60000</td>
<td>0.50847</td>
<td>0.55046</td>
</tr>
<tr>
<td>Character-Embeddings</td>
<td>Full</td>
<td>0.62595</td>
<td>0.43501</td>
<td>0.51330</td>
</tr>
<tr>
<td>Character-Embeddings</td>
<td>Filtered</td>
<td>0.65341</td>
<td>0.48729</td>
<td>0.55825</td>
</tr>
<tr>
<td>Mimic</td>
<td>Full</td>
<td>0.58233</td>
<td>0.38462</td>
<td>0.46326</td>
</tr>
<tr>
<td>Mimic</td>
<td>Filtered</td>
<td>0.61165</td>
<td>0.53390</td>
<td><strong>0.57014</strong></td>
</tr>
</tbody>
</table>

Table 1: Experimental Results

<table>
<thead>
<tr>
<th>System</th>
<th>Data</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virk et al</td>
<td>Full</td>
<td>35.6</td>
</tr>
<tr>
<td>Character-Embedd</td>
<td>Filtered</td>
<td>45.9</td>
</tr>
<tr>
<td>Virk et al</td>
<td>Full</td>
<td>52.9</td>
</tr>
<tr>
<td>Character-Embedd</td>
<td>Filtered</td>
<td><strong>62.9</strong></td>
</tr>
</tbody>
</table>

Table 2: Comparison to a previously reported system

9 Conclusions and Future Work

Our main contributions are two-fold. First, we have reported a deep learning based system for the automatic extraction of typological information from descriptive grammars of natural languages. As mentioned previously, the manual extraction of such information is very costly both in terms of cost and human efforts. Any assistance in this regard is much appreciated as typological linguistic information is not only useful for investigating the linguistic diversity of the universe, but is also being used for many other NLP related tasks. A survey of usefulness of typological information in various NLP tasks can be found in (O’Horan et al., 2016).

Unlike, previously reported systems for the same task, the system proposed in this study uses word-embeddings as the only knowledge source and does not require any feature engineering to identify suitable feature set for the machine learning part.

Second, we have shown how word-embeddings learned from general purpose text can be used in a domain specific setting, and how character embeddings can also be used as a work around for out-of-vocabulary terms. Further, inferring word embeddings is another way to deal with out of vocabulary words as long as the words are not cross-lingual (English and non-English in our case). In the future, we would like to improve the system by experimenting with n-gram embedding instead of character embeddings. In another direction, word-embeddings could be learned from domain-specific data-sets as opposed to general purpose word-embeddings used in this study, which could avoid the issue of out-of-vocabulary words issue.

Evaluation of the extracted typological information is another task that we have plans to carry out in the near future. One can select a suitable set of features from one of the existing typological databases, and use their values as a gold-standard to evaluate the performance of the system proposed in this study.

Acknowledgements

The work presented here was funded by (1) the Dictionary/Grammar Reading Machine: Computational Tools for Accessing the World’s Linguistic Heritage (DReaM) Project awarded 2018–2020 by the Joint Programming Initiative in Cultural Heritage and Global Change, Digital Heritage and Riksantikvarieämbetet, Sweden; (2) the Swedish Research Council as a part of the project South Asia as a linguistic area? Exploring big-data methods in areal and genetic linguistics (2015–2019, dnr 421-2014-969); (3) From Dust to Dawn: Multilingual Grammar Extraction from Grammars project funded by Stiftelsen Marcus och Amalia Wallenberg’s Minnesfond 2007.0105, Uppsala University; (4) the Swedish Research Council as part of the project Evaluation and refinement of an enhanced OCR-process for mass digitisation (2019–2020, dnr IN18-0940:1 and 421-2014-969). It is also supported by Språkbanken Text and Swe-CLARIN, a Swedish consortium in Common Language Resources and Technology Infrastructure (CLARIN) Swedish CLARIN (dnr 821-2013-2003).
References


Wichmann Søren and Taraka Rama. 2019. Towards unsupervised extraction of linguistic typological features from language descriptions. In First Workshop on Typology for Polyglot NLP, Florence.


A Algorithm 1

1: procedure EXTRACT_ADJECTIVE-NOUN_ORDER(parse)
2: for <every frame in parse> do
3: if frame = SEQUENCE then
4:   NA ← False
5:   AN ← False
6:   Both ← False
7:   if 'adjective' ∈ Entity_1 ∧ 'noun' ∈ Entity_2 then
8:     if Frequency ∈ [sometimes, usually, mostly, often] then
9:       Both ← True
10:      else if order = follow then
11:       AN ← True
12:      else if order = precede then
13:       NA ← True
14:    end if
15: end if
16: end if
17: end for
18: end procedure
## B  Extracted Typological Information

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject and NP Order</td>
<td>NP–SubjectMarker</td>
</tr>
<tr>
<td>Object and NP Order</td>
<td>NP–ObjectMarker</td>
</tr>
<tr>
<td>Constituent Order</td>
<td>SOV</td>
</tr>
<tr>
<td>PostpositionalPhrase–Oblique-markedNP Order</td>
<td>Both</td>
</tr>
<tr>
<td>ObliguePhrase–SubjectOFClause Order</td>
<td>SubjectOFClause–ObliquePhrase</td>
</tr>
<tr>
<td>ObliguePhrase–Verb Order</td>
<td>ObliguePhrase–Verb</td>
</tr>
<tr>
<td>Negator–Verb Order</td>
<td>Negator–Verb</td>
</tr>
<tr>
<td>AdPosition–NP Order</td>
<td>NP–AdPosition</td>
</tr>
<tr>
<td>Possessor–Possessum Order</td>
<td>Possessor–Possessum</td>
</tr>
<tr>
<td>Adjective–Noun Order</td>
<td>Noun–Adjective</td>
</tr>
<tr>
<td>Demonstrative–Noun Order</td>
<td>Noun–Demonstrative</td>
</tr>
<tr>
<td>Numeral–Noun Order</td>
<td>Noun–Numeral</td>
</tr>
<tr>
<td>RelativeClause–HeadNoun Order</td>
<td>RelativeClause–HeadNoun</td>
</tr>
<tr>
<td>PossessivePronoun–Noun Order</td>
<td>PossessivePronoun–Noun</td>
</tr>
<tr>
<td>ObliqueMarker–Noun Order</td>
<td>Noun–ObliqueMarker</td>
</tr>
<tr>
<td>TransitiveVerb–ObjectMarker Order</td>
<td>TransitiveVerb–ObjectMarker</td>
</tr>
<tr>
<td>NominalizedVerb–SubjectMarker Order</td>
<td>SubjectMarker–NominalizedVerb</td>
</tr>
<tr>
<td>Verb–DirectObject Order</td>
<td>DirectObject–Verb</td>
</tr>
<tr>
<td>Oblique–Verb Order</td>
<td>Oblique–Verb</td>
</tr>
<tr>
<td>Oblique–Subject Order</td>
<td>Subject–Oblique</td>
</tr>
<tr>
<td>Adverb–Subject Order</td>
<td>Subject–Adverb</td>
</tr>
<tr>
<td>Adverb–Object Order</td>
<td>Adverb–Object</td>
</tr>
<tr>
<td>Adverb–Oblique-markedNP Order</td>
<td>Adverb–Oblique-markedNPs</td>
</tr>
<tr>
<td>NasalSegments–VoicelessStops Order</td>
<td>NasalSegments–VoicelessStops</td>
</tr>
<tr>
<td>LabialNasal–PalatoAlveolar Order</td>
<td>LabialNasal–PalatoAlveolar</td>
</tr>
<tr>
<td>HomorganicNasals–VoicelessStops Order</td>
<td>HomorganicNasals–VoicelessStops</td>
</tr>
<tr>
<td>Liquids–LabialStops Order</td>
<td>LabialStops–Liquids</td>
</tr>
</tbody>
</table>
Recognizing and Splitting Conditional Sentences for Automation of Business Processes Management

Ngoc Phuoc An Vo, Irene Manotas, Octavian Popescu, Algimantas Ėcerniauskas*, Vadim Sheinin
IBM Research, Yorktown Heights, US
*IBM Client Innovation Center Baltic, Vilnius, Lithuania
{ngoc.phuoc.an.vo, irene.manotas, algimantas.cerniauskas}@ibm.com
{o.popescu, vadims}@us.ibm.com

Abstract

Business Process Management (BPM) is the discipline which is responsible for management of discovering, analyzing, redesigning, monitoring, and controlling business processes. One of the most crucial tasks of BPM is discovering and modelling business processes from text documents. In this paper, we present our system that resolves an end-to-end problem consisting of 1) recognizing conditional sentences from technical documents, 2) finding boundaries to extract conditional and resultant clauses from each conditional sentence, and 3) categorizing resultant clause as Action or Consequence which later helps to generate new steps in our business process model automatically. We created a new dataset and three models to solve this problem. Our best model achieved very promising results of 83.82, 87.84, and 85.75 for Precision, Recall, and F1, respectively, for extracting Condition, Action, and Consequence clauses using Exact Match metric.

1 Introduction

Business processes which serve as basis for many (if not all) companies are usually stored as unstructured data, especially as text documents (Blumberg and Atre, 2003). They can reflect all organization tasks and activities in order to provide services. Therefore, these documents could be converted into process models which allows to discover, analyze, redesign, monitor and control these business processes (M. Dumas, 2018). The discipline which is responsible for management of this life cycle is known as Business Process Management (BPM).

Today, this well known practice is very interdisciplinary and combines: organizational theory, management, and even computer science. One of the sub-areas of computer science is Natural Language Processing (NLP) and due to large number of freely available NLP tools such as taggers, parsers, and the lexical database WordNet, NLP is applied in a multitude of domains and BMP is no exception. The application of NLP techniques could be used to extract useful information from textual documents to infer a process model, and also to extract information from the process model to facilitate visual process analysis (Leopold, 2013).

Process modelling is a very complex and time-consuming task. In this work we mainly focused on the analysis of extracted information from text documents in order to have more accurate process models. Given a list of extracted sentences from technical documents, the contributions of our work consists of the following: 1) classifying if a sentence is a conditional sentence, 2) identifying and extracting the conditional and resultant clauses from a conditional sentence, and 3) categorizing the extracted resultant. Section 2 introduces business context and application for our tasks. Section 3 explains how we defined the tasks and created the new dataset TechCond towards solving the tasks. Section 4 presents our methods and models for the tasks. Section 5 shows experiments, evaluation results, and the error analysis that identifies the limitations of our models.

2 Business Context, Application, and Related Works

Process Discovery Traditional process models based on interviews often provide only a limited or biased picture of the actual process. Such highly abstract result can be interpreted in many ways. Therefore, during the past decade process mining and visualization tools (i.e., Celonis, UIPath) were developed to improve process visibility by generating highly adaptable, highly maintainable and validated business process models. However, these tools focus only on analysis of results from structured data - process sequences from actual user
Desktops, or systems logs. It is well known that Desktop Procedures (DTPs) and Standard Operation Procedures (SOP) are very common process documentation techniques (Phalp and Cox, 2007). Therefore, in order to see a broader picture of the process the information from textual source should be analyzed and included into process model. One of the tools which includes multiple sources is Process Discovery Accelerator (PDA).

**Application**  PDA is a business process mining and discovery tool currently available only for automation consultants of our company. It runs as a web application in production on the Services Essentials for Automation platform of our company. PDA has three main components: 1) Standard operation procedure file analysis; 2) Click stream file analysis, sequential structured data which represents actual users activities; and 3) system logs file analysis which is also structured data from applications (system, e-mail, chat). Process Discovery Accelerator is able to discover processes using separate source components or all available at once, this procedure is called process harmonization. The final view of this procedure is single, correlated or uncorrelated processes from all available sources. The example of a single (only SOP) simple process graph result in PDA UI is shown in Figure 1. It illustrates how process flow looks like when conditional and resultant clauses are split into separate steps.

The analysis of Standard operation procedures is very challenging not only because it is based on unstructured data but also because it is very non-homogeneous, domain specific and biased since it is highly dependent on a process description creator (Baier and Mendling, 2013). These SOP files usually contain complex step descriptions, i.e., “if something, then do this” (Table 1) which sometimes could act as gateways or decision points. In order to have a more accurate process model we need to detect specific clause (conditions and actions) from already extracted process steps.

The main objective of PDA is to find automation opportunities by analyzing available data from Standard operation procedures, Click stream, and log files. Without accurate analysis of SOP files it is difficult to measure differences between different sources. Our proposed solution for conditional splitting is crucial for the SOP analysis, since it allows to see actual decision points and how complex a process is by detecting actions and/or consequences.

**Related Works**  The study (Wenzina R., 2013) proposed heuristic, rule-based method to identify condition-action sentences. By using a set of linguistic patterns authors were able to split up sentences semantically into their clauses showing the condition and the consequence with recall of 75% and precision of 88%. Note: these numbers are for specific patterns - for all activities recall is 56%. Another rule-based work was presented by (Hematialam H., 2017). This work is based on combinations of part of speech tags used as features. Authors applied more statistical approach to automatically extract features, however, the recall value is very similar to (Wenzina R., 2013). Both works were tested using sentences from medical SOP’s.

Another good example of NLP and Business Process Management combination is the work from (Qian et al., 2019). Authors used latest NLP techniques (transformers) to classify textual information from word-level to sentence-level. Although this work is most similar to our work it focuses more on automatic process graph extraction, which aims to extract multi-granularity information without manually defined procedural knowledge.

### 3 Task Definition and New Dataset

In this section, we first analyzed the sampled data from our client, then we defined the task Conditional Sentence Splitting, and created a new dataset TechCond to solve it.

#### 3.1 Preliminary Data Analysis

We received a dataset of 4,098 sampled sentences provided by our client from SOP documents that is based on few industries: financial services (banking) and retail. This dataset was manually annotated by client in four categories (at sentence level) with unbalanced label distribution:

- **No Condition (NC) (86.5%)**: sentence that has no conditional logic, no condition is found.
- **Condition Action (CA) (9%)**: sentence that has condition, and resultant is an Action.
- **Condition Consequence (CC) (3%)**: sentence that has condition, resultant is a Consequence.

[1]https://www.foodandbeveragetrainer.com/sop/

1495
Figure 1: Standard operation procedure example of welcoming guest in restaurant from food and beverage training in Process Discovery Accelerator tool UI. The process step in diamond shape indicates condition clause and dashed arrow shows resultant. Solid lines connects different process steps.

Figure 2: Syntactic structures of CA sentences.

- Only Condition (OC) (1.5%): sentence that has only condition, no resultant found.

We extracted 352 CA sentences from this client’s dataset then analyzed syntactic structures of CA sentences and different types of Action resultants (Figures 2 and 3). Based on this preliminary analysis, we then created the new dataset TechCond (Section 3.3) which has better label distribution and more fine-grained annotation. We also constructed our rule-based model (Section 4) based on the analysis of syntactic structures and Action resultant types of CA sentences.

Figure 3: Action resultant types of CA sentences.

3.2 Task Definition

We defined the end-to-end task Conditional Sentence Splitting in which the input is a given sentence, and the output is the extracted clauses classified into Condition and other resultant categories such as Action or Consequence. Since this is a large and complex task, it can be split into the following three subtasks.

Subtask 1: Conditional Sentence Classification

This task recognizes whether a given sentence is in conditional form. Conditional sentence is a sentence that expresses one or more thing(s) contingent on something else. The most basic form of a conditional sentence is

\[ \text{If } P, Q \]  

(1)
where $P$ is the conditional clause, and $Q$ is the resultant. Recognizing conditional sentences is not a trivial task because there are various ways of expressing the meaning in conditional sentences. A basic approach to identify conditional sentences is to search for a conditional clause within a sentence that has keywords like *If* or *Unless* as condition indicators. However, conditional sentences do not always appear in standard conditional form in equation (1) with these indicators. Table 1 shows examples of conditional sentences that appear in different ways.

Subtask 2: Sentence Boundary Finding and Splitting This task finds the boundary of conditional and resultant clauses in a conditional sentence. A basic approach is to locate the comma that separates between conditional and resultant clauses. For example: *If it rains, children stay inside.* The comma in this example is a strong indicator showing that the first part of the sentence that has condition indicator *If* is a conditional clause, and the second part is a resultant. However, as conditional sentences can appear in numerous ways (Table 1), this makes conditional sentence splitting a complex task.

Subtask 3: Resultant Categorization After splitting a conditional sentence into conditional and resultant clauses, the next step is to categorize if the resultant is an Action or a Consequence clause. In our application, we are interested in Action clause as it helps us to create a new action branch for our process model. For example:

- **Refer to the author if you are in any doubt about the currency of this document.** The resultant "Refer to the author" is an Action clause which means the action will be taken should the condition be satisfied.

- **If the entered password is matched with the one stored in system, the user is authenticated.** The resultant "the user is authenticated" is a Consequence clause which shows the result should the condition be satisfied. Thus, no action is captured and no action branch is created for our process model.

Some common types of Action resultant are:

- Imperative type is phrase that has an imperative verb to give a command or an order.
- Obligation type is phrase that expresses something necessary to follow or execute. It usually has modals of obligation such as *must, have to, have got to, need to, etc.*
- Others are phrases that are neither in imperative nor in obligation form but actual meaning expresses the same.

3.3 TechCond Dataset
We created the new TechCond dataset with conditional sentences to predict sentences with conditions and different types of resultants. We extracted sentences from real text documents collected from public technical manuals and regulation documents from different websites. Technical documents included manuals and troubleshooting guides for different software products and applications available online as documents that could be downloaded. These documents were processed by the English Slot Grammar (ESG) parser (McCord et al., 2012) to identify sentences and their linguistic features. We filter out sentences having conditional words and a subordinating conjunction in their constituency parse. We tried to reduce the bias towards a specific type of conditional sentences by considering a diverse set of conditional words: *'after', 'as a consequence of', 'as a result of', 'assuming', 'if', 'if only', 'if not', 'only if', 'unless', 'until', 'when', 'because', 'as soon as', 'as long as', 'but for', 'even if', 'once', 'on the condition', 'provided', 'providing*. After we filtered out sentences potentially describing a condition, a total of 1,936 sentences with conditional statements were extracted for annotation to be part of the TechCond dataset.

Annotation Methodology We selected the Doccano \(^2\) annotation tool to annotate sequences in the sentences according to our task. Guidelines for the annotation job included possible types of sentences and clauses to annotate. The labels included in the annotation were the following:

- **Condition (CD):** Clause describing a condition. A conditional clause can come before or after the main clause.
- **Action (AC):** Clause describing an actionable result that is related to a condition. The condition can be described before or after the Action clause in the same sentence.

---

\(^2\)https://doccano.herokuapp.com/
Conditional Sentences Actual Meanings
1 If it rains, children should stay home. If it rains, children should stay home.
2 Unless it rains, children can go out. If it does not rain, children can go out.
3 1. Otherwise, they can go out. If it does not rain, children can go out.
4 Come now and I’ll give you the book. If you come now, I’ll give you the book.
5 Do you like it? You can have it now. If you like it, you can have it now.
6 For rainy days, children stay home. If it rains, children stay home.
7 Anyone who skips class will be disciplined. If anyone skips class, they will be disciplined.

Table 1: Examples of Conditional Sentences appear in various ways.

<table>
<thead>
<tr>
<th>Data</th>
<th>CD</th>
<th>CS</th>
<th>AC</th>
<th>OC</th>
<th>NC</th>
<th>UA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>1,446</td>
<td>568</td>
<td>886</td>
<td>18</td>
<td>26</td>
<td>83</td>
</tr>
<tr>
<td>Test</td>
<td>184</td>
<td>71</td>
<td>114</td>
<td>3</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>Dev</td>
<td>184</td>
<td>67</td>
<td>114</td>
<td>3</td>
<td>5</td>
<td>23</td>
</tr>
<tr>
<td>Total</td>
<td>1,814</td>
<td>706</td>
<td>1,114</td>
<td>24</td>
<td>35</td>
<td>122</td>
</tr>
</tbody>
</table>

Table 2: Frequency of labels by data split.

- Consequence (CS): Clause describing a non-actionable activity i.e., it does not describe an action that a system or user needs/should take.
- Only-Condition (OC): Sentence describing only a condition without additional clauses for actions or activities.
- No Condition (NC): Sentence with no conditional logic.
- Unconditional-Action: Clause describing only an action without an associated condition.

Annotators were guided to use labels at the sentence or clause level. *Sentence-level* labels included No-Condition (NC) and Only-Condition (OC) labels, and can only be assigned to a whole sentence. *Clause-level* labels included Condition (CD), Action (AC), and Consequence (CS) labels can be assigned only to a clause in a sentence having at least two clauses.

Before the annotation task, annotators were guided to analyze patterns of conditional sentences that could be found during the annotation task. For instance, sentences with short phrases such as *if possible* or *if any*, were not considered conditional phrases. Three annotators completed the annotation job for 1,936 sentences. Since a sentence having a condition can have a resultant of type consequence or action, there could be sentences with two or more types of labeled sequences. Table 2 shows the different data splits of the TechCond dataset with the final number of annotated sequences per label.

4 Model Descriptions

We present three models using both rule-based and deep learning approaches to recognize and split conditional sentences into conditional clause and other resultant categories. The rule-based model implements a pipeline that solves the end-to-end task by solving each subtask. In contrast, we considered this large task as a sequential labeling task for deep learning approach.

**Rule-Based Model**

Syntactic and semantic manually written rules are the basis for many systems that analyze technical documents, for example (Panapikolaou, 2012; Zhang and El-Gohary, 2012). In a nutshell, these approaches adopt a general architecture, which extracts the constituents of the sentence, their type and their dependency relationships using constituency or dependency parsers. Among the extracted entities, there could be conditional constituents which are further analyzed. However, we have found out that for our corpus of technical documents the systems trained on standard corpora, (Marcus et al., 1993; Taylor et al., 2003) do not perform well. A source of error might be the text itself, as the technical corpora is not necessarily written by native English speakers, but most importantly, conditional clauses have many peculiarities, both at syntactic and semantic level, which could be exploited more efficiently than a general approach does.

To achieve a high quality of parsing of the conditional sentences, we developed a system of rules that proceeds sequentially to determine (i) the conditional triggers - using lexical and semantic rules (ii) boundaries of conditional clause - using syntactic and semantic rules, (iii) the border of adjacent
constituents - using syntactic rules and (iv) the relationship between constituents and the corresponding conditional - using semantic rules.

The conditional triggers are lexical items, like *if*, *when*, but also the comma itself, “,”, “,”, or *that*, when it introduces a relative clause. The rules in (i) analyze the context and decide whether a trigger does indeed introduce a conditional. Then, the one predicate condition is used by the rules in (ii), that is, we determine exactly one verbal phrase whose boundaries are actually the boundaries of the conditional clause itself. We used semantic features, like *imperative*, or *nominalization* to decide whether a verbal group carries conditional relevance. Some of these features are extracted directly from ESG output (McCord et al., 2012), and some other are extracted by our rules via a deep analysis of copula, semantic transparent verbs and other verbal constructions that do not necessarily contain exactly one verb. The boundaries ambiguities are resolved by rules in (iii), which decide whether a noun phrase is a part of a conditional or it belongs to the following clause. The rules in (iv) check for imperative, or imperative like constructions, because a clause expressing an action that logically follows a conditional must be understood as “to do” obligation. That is, these rules rank the candidates according to their imperative valency and their position with respect to the determined conditional clause.

This model achieved an accuracy of about 70% (Table 3).

**BERT-based Model** One sequence model for conditional splitting was based on the *BERT*BASE language model (Devlin et al., 2019). This model takes a maximum 512 input word piece token sequence $X = [x_1; x_2; \ldots; x_T]$ and uses a $L = 12$ layer transformer network, with 12 attention heads and 768 embedding dimensions, to output a sequence of contextualized token representations. We used the representation of the first sub-token as the input to the token-level classifier over the conditional label set. We fine-tuned the model using 5 epochs, learning-rate 0.1, and 256 maximum sequence length.

**XLM-R-based Model** XLM-RoBERTa (XLM-R) is a SOTA multilingual masked language model trained on 2.5 TB of newly created clean CommonCrawl data in 100 languages (Conneau et al., 2019). It obtains strong gains over previous multilingual models like mBERT and XLM on classification, sequence labeling and question answering. We used the XLM − R model with standard settings: max sequence 256, learning rate 7e-5, warmup_proportion 0.1, epoch 5, batch size 8.

## 5 Experiments

To evaluate the performance of the models, we used the test set from the TechCond dataset (Section 3.3) following the data split shown in Table 2. Since our dataset was annotated at phrase/chunk level, we transformed our annotated data into Inside-Outside-Beggining (IOB) tagging format for the sequential labeling task. For example:

- **Phrase level annotation:** { "id": 908, "text": "Include the date if the opt-out period expires.”, "meta": , "annotation_approver": "admin", "labels": [0, 16, "Action"], [17, 47, "Condition"]}

- **IOB annotation scheme:** {Include B-Action, the I-Action, date I-Action, if B-Condition, the I-Condition, opt-out I-Condition, period I-Condition, expires I-Condition, . O}

Since we are only interested in having sentences split into three main categories Condition, Action, and Consequence, we focused the evaluation of the models for these labels. Table 3 shows the performance of our models for the end-to-end conditional sentence splitting on the test set of 200 sentences. The XLM-R-based model outperforms other two counterparts by large margins.
Error Analysis Figure 4 shows sentences having complex structures that our models failed to recognize and extract correct Condition, Action, and Consequence clauses. Although the rule-based model has a comprehensive set of rules for high accuracy, the coverage is limited for sentences having complex syntactic structures. It struggles to capture non-standard conditional structures, such as relative clause where no Condition clause was found (Example 1); or it captured wrong resultant (Example 2) due to structure complexity. In contrast, deep learning models performed well on (Examples 1 & 2) but failed to capture correct resultant (Example 3) due to the coordination of verbs in the Condition clause; or failed to capture full resultant (Example 4) due to a very long sentence sequence. Our models also have difficulty to process sentences with multiple conditions or multiple resultants. For example:

- **If you had dynamic SQL** (or if you re-bound static SQL), your applications might be breached.

- **If using PayPal for payment then click on the PayPal tab and then click Pay Now.**

Example 3 shows another limitation that is to differentiate between Unconditional-Action clause "Verify in MSR link the order quantity" and the actual Action clause.

6 Conclusions and Future Work Recognizing and splitting conditional sentences for automation of business process management is a complex and important task for many industries. We presented a non-trivial real-world system that can recognize and split technical instructions (at sentence level) into Condition, Action, Consequence clauses for business process modeling and automatic update. We defined an end-to-end task consisting of three subtasks that fits our business context and needs. We also created a new dataset and three models, using both rule-based and deep learning approaches, to solve this task. For future work, we plan to improve the task performance by 1) expanding the dataset and improving annotation to capture more examples of complex structures, and 2) trying better learning approaches or more linguistics attributes to help our models learn better.

References


“Don’t discuss”: Investigating Semantic and Argumentative Features for Supervised Propagandist Message Detection and Classification

Vorakit Vorakitphan, Elena Cabrio, Serena Villata
Université Côte d’Azur, Inria, CNRS, I3S, France
vorakit.vorakitphan@inria.fr, elena.cabrio@univ-cotedazur.fr, villata@i3s.unice.fr

Abstract

One of the mechanisms through which disinformation is spreading online, in particular through social media, is by employing propaganda techniques. These include specific rhetorical and psychological strategies, ranging from leveraging on emotions to exploiting logical fallacies. In this paper, our goal is to push forward research on propaganda detection based on text analysis, given the crucial role these methods may play to address this main societal issue. More precisely, we propose a supervised approach to classify textual snippets both as propaganda messages and according to the precise applied propaganda technique, as well as a detailed linguistic analysis of the features characterising propaganda information in text (e.g., semantic, sentiment and argumentation features). Extensive experiments conducted on two available propagandist resources (i.e., NLP4IF’19 and SemEval’20-Task 11 datasets) show that the proposed approach, leveraging different language models and the investigated linguistic features, achieves very promising results on propaganda classification, both at sentence- and at fragment-level.

1 Introduction

Propaganda represents an effective, even though often misleading, communication strategy to promote a cause or a viewpoint, for instance in the political context (Lasswell, 1938; Koppang, 2009; Dillard and Pfau, 2009; Longpre et al., 2019). Different communication means can be used to disseminate propaganda, i.e., textual documents, images, videos and oral speeches. The ability to effectively identify and manifestly label such kind of misleading and potentially harmful content is of primarily importance to restrain the spread of such information to avoid detrimental consequences for the society.

In this paper, we tackle this challenging issue (Da San Martino et al., 2020b) by proposing a textual propaganda detection model. More precisely, we address the following research questions: (i) how to automatically identify propaganda in textual documents and further classify them into fine-grained categories?, and (ii) what are the linguistic distinctive features of propaganda text snippets? The contribution of this paper consists not only in proposing a new effective neural architecture to automatically identify and classify propaganda in text, but we also present a detailed linguistic analysis of the features characterising propaganda messages.

Our work focuses on the propaganda detection and classification task, casting it both as a binary and as a multi-class classification task, and we address it both at sentence-level and at fragment-level. We investigate different architectures of recent language models (i.e., BERT, RoBERTa), combining them with a rich set of linguistic features ranging from sentiment and emotion to argumentation features, to rhetorical stylistic ones. The extensive experiments we conducted on two standard benchmarks (i.e., the NLP4IF’19 and SemEval’20-Task 11 datasets) show that the proposed architectures achieve satisfying results, outperforming state-of-the-art systems on most of the propaganda detection and classification subtasks. An error analysis discusses the main sources of misclassification. Furthermore, we analysed how the most relevant features for propaganda detection impact the fine-grained classification of the different techniques employed in propagandist text, revealing the importance of semantic and argumentation features.

2 Related Work

In the last years, there has been an increasing interest in investigating methods for textual propaganda detection and classification. Among them, (Barrón-
Cedeño et al., 2019) present a system to organize news events according to the level of propagandist content in the articles, and introduces a new corpus (QProp) annotated with the propaganda vs. trustworthy classes, providing information about the source of the news articles. (Da San Martino et al., 2019) present the benchmark of the shared task NLP4IF'19 on fine-grained propaganda detection. As a follow up, in 2020 SemEval proposed a shared task (T11) (Da San Martino et al., 2020a) reducing the number of propaganda categories with respect to NLP4IF'19, and proposing a more restrictive evaluation scheme. To evaluate the proposed approach, we rely on these two standard benchmarks, i.e., the NLP4IF'19 and SemEval'20 datasets.

The most recent approaches for propaganda detection are based on language models that mostly involve transformer-based architectures. The approach that performed best on the NLP4IF'19 sentence-level classification task relies on the BERT architecture with hyperparameters tuning without activation function (Mapes et al., 2019). (Yoosuf and Yang, 2019) focused first on the pre-processing steps to provide more information regarding the language model along with existing propaganda techniques, then they employ the BERT architecture casting the task as a sequence labeling problem. The systems that took part in the SemEval 2020 Challenge - Task 11 represent the most recent approaches to identify propaganda techniques based on given propagandist spans. The most interesting and successful approach (Jurkiewicz et al., 2020) proposes first to extend the training data from a free text corpus as a silver dataset, and second, an ensemble model that exploits both the gold and silver datasets during the training steps to achieve the highest scores. Notice that most of the most performing recent models heavily rely on transformer-based architectures.

In this paper, we also rely on language model architectures for the detection and classification of propaganda messages, empowering them with a rich set of features we identified as pivotal in propagandist text from the computational social science literature. In particular, (Morris, 2012) discusses how emotional markers and affect at word- or phrase-level are employed in propaganda text, whilst (Ahmad et al., 2019) show that the most effective technique to extract sentiment for the propaganda detection task is to rely on lexicon-based tailored dictionaries. Recent studies (Li et al., 2017) show how to detect degrees of strength from calmness to exaggeration in press releases. Finally, (Troiano et al., 2018) focus on feature extraction of text exaggeration and show that main factors include imageability, unexpectedness, and the polarity of a sentence.

3 Propaganda Detection as a Classification Task

(Da San Martino et al., 2019) define the Fine-Grained Propaganda Detection task as two sub-tasks, with different granularities: i) Sentence-Level Classification task (SLC), which asks to predict whether a sentence contains at least one propaganda technique, and ii) Fragment-Level Classification task (FLC), which asks to identify both the spans and the type of propaganda technique.

In the following example, “In a glaring sign of just how stupid and petty things have become in Washington these days, Manchin was invited on Fox News Tuesday morning to discuss how he was one of the only Democrats in the chamber for the State of the Union speech not looking as though Trump killed his grandma.” the span “stupid and petty” carries some propagandist bias, and is labeled as “Loaded Language”, “not looking as though Trump killed his grandma” is considered as “Exaggeration and Minimisation”, and “killed his grandma” is “Loaded Language”. According to the SLC task, the whole sentence should be classified as a propaganda message given that it contains at least one token (e.g., “stupid and petty”) considered as such.

As previously introduced, current systems address these tasks relying on word embedding models (e.g., BERT-embedding) and standard features (e.g., PoS, name-entity, n-grams), as representations to feed various RNN architectures (Morio et al., 2020; Chernyavskiy et al., 2020). Recently, the language model BERT (Devlin et al., 2019) has been widely utilized to optimize the performances of classification tasks, but there is still room for improvement, in particular when applied to propaganda detection (Da San Martino et al., 2020a, 2019). In this work, we experiment with multiple architectures and language models to classify propagandist messages on both sentence and fragment-level. Prior to that, we conduct a detailed investigation of linguistic and argumentation features to capture propaganda strategies.
4 Feature Analysis

Propaganda strategies generally involve specific targets to be stimulated by the message. To better study such techniques from a computational point of view, we investigate a set of features that we assume to play a role in propaganda.

4.1 Persuasion

Speech style. To analyze the writing style of the messages, we apply the dictionary-based mapping tool “General Inquirer (v. 1.02)” (Gilman, 1968). It relies on a list of lexicons from 26 domains (e.g., politician speeches, consumer protests) annotated according to 182 rating categories and dimensions (e.g., valence categories and words indicating overstatement and understatement)\(^2\). We apply such a tool on our data and we sum the ratings of each token to obtain a global score for a sentence.

Lexical complexity. Given that pre-trained language models have shown to capture lexical and semantic complexities of words, we rely on BERT (base-uncased) (Devlin et al., 2019) to extract lexical complexity features. We extract a vector of 768 dimensions per each token, then we average w.r.t. all tokens in a sentence, to obtain one vector of 768 dimensions to represent a sentence.

Concreteness. Propaganda messages tend to employ words with concrete meaning, that has more impact in conveying the intention of the message than using abstract words (Eliasberg, 1957). We rely on the concreteness lexicon (Brysbaert et al., 2013) and we sum the standardized score of each token in a sentence to obtain the global score.

Subjectivity. We rely on the subjectivity lexicon from (Wilson et al., 2005). We sum up the scores over all tokens in a sentence found in the lexicon as our extracted feature. Each word labeled as “weaksubj” is set to 0.5, and “strongsubj” to 1.

4.2 Sentiment

Sentiment labels. We use SentiWordNet 3.0 (Baccianella et al., 2010) to obtain word-level sentiment labels (positive, negative, or neutral). We sum the sentiment scores of each word in a sentence, producing a vector with 3 dimensions (i.e., pos, neg, neu) for each sentence.

Emotion labels. We extract 8 emotions (i.e., afraid, amused, angry, annoyed, don’t care, happy, inspired, sad) from DepecheMood++ lexicon (Araque et al., 2019). For each word that evokes emotions in a sentence, we produce our features by summing up each set of emotions evoked by each token, then find the average by emotions. Hence, we produce 8 emotion scores for a sentence.

VAD labels. In the three-dimensional model of affect, valence ranges from unhappiness to happiness and expresses the pleasant or unpleasant feeling about something, arousal expresses the level of affective activation, ranging from sleep to excitement, and dominance reflects the level of control of the emotional state, from submissive to dominant. We use Warriner lexicon (Warriner et al., 2013) to match each word in a sentence to its VAD standardized word scores and sum up as our features.

Connotation. Propaganda can convey sentiment beyond its original meaning. Connotation lexicon (Feng et al., 2013) provides positive, negative and neutral labels of each word. We count the frequencies of the three labels evoked in each sentence.

Politeness. Politeness evokes sentiment in readers. We use a lexicon of positive and negative words from (Danescu-Niculescu-Mizil et al., 2013), then we count the frequencies of both positive and negative words found in each sentence.

4.3 Message Simplicity

To keep the message simple and picturable is one of main purposes of propaganda. We list the features we considered to extract the simplicity of message.

Exaggeration. We use imageability lexicon (Tsvetkov et al., 2014) based on picturable vocabulary which mentally leads to an exaggerating state of mind. We consider the scores of abstraction and concreteness at each word token. We then sum up the scores for all the labels found in a sentence.

Length. “The less words used, the better to understand” can be a concept to easily interpret the propagandist message. We apply two strategies: \(^i\) we count the average char-length, actual char-length, word length, punctuation frequency, capital-case frequency per sentence (Ferreira Cruz et al., 2019); \(^ii\) we apply length encoding at character-level, plus one additional dimension for non-alphabetical char count.

Pronouns. Loaded language, name calling and labelling are the most used techniques in propaganda text (Da San Martino et al., 2019), and they all make use of pronouns. We create a lexicon of 123 pronouns in English\(^3\) and perform one-hot encoding.
encoding of common used pronouns in English.

4.4 Argumentation
We assume that argumentation plays an important role in propaganda. To extract argumentative features representing our data, we train a supervised classifier for the task of argumentative sentence classification on the persuasive essays dataset (Stab and Gurevych, 2014). First, we cast it as a binary classification task, merging premises, claims and major claims into the argumentative label, as opposed to the non-argumentative label. Then, for the argumentation component task, we rearrange the data to binary labels where the major claims and claims labels are merged, and they are opposed to premises. To address these tasks, we build and fine-tune a BERT classifier. We use a learning rate of 1e-5 with 80/20 split of the dataset. We run our classifier 3 times at different random states. The results for the argumentative sentence classification are (macro-average) F1 0.84, precision 0.86, recall 0.82, while for the component classification they are F1 0.77, precision 0.80, recall 0.75.

To extract argumentative features from the annotations provided by our classifiers, we use BERT-based features. After fine-tuning, we freeze the hidden states of these fine-tuned BERT models. To extract the argumentative and components features from each classifier, we take the [CLS] token of each sentence from the fine-tuned BERT model.

4.5 Ablation Study
To investigate the impact of the proposed features (Section 4) for propaganda detection, we perform ablation tests by testing a supervised classifier relying on BERT + logistic regression. To the purpose, we use the NLP4IF’19 training and test sets (Da San Martino et al., 2019).

Table 1 reports on the performances obtained while integrating groups of features to the proposed model. A logistic regression model is used as a baseline. Best results are obtained when adding all the proposed features, but the argumentation ones. Argumentation features alone perform almost identical as semantic features, therefore - unexpectedly - no added value can be demonstrated.

5 Sentence-level Classification
In the following, we describe the experiments we carried out to address the propaganda detection task at sentence level, investigating different architectures and leveraging both recent language models and the features that proved to play a role in propaganda messages. For the evaluation, we used the two available datasets for propaganda detection: i) the NLP4IF’19 data set (Da San Martino et al., 2019) (293 articles for training and 101 for testing); and ii) the data from SemEval’20 T11 (Da San Martino et al., 2020a) (371 articles for training and 75 in the development set).

5.1 Prediction Models
In the following, we first describe the baseline and the SOTA models we tested in our experiments, and then we present the three architectures we propose (underlined) integrating the propagandist features previously investigated (Section 4).

BERT. Our baseline model relies on a pre-trained bidirectional transformer language model to encode context specific sentence tokens (Devlin et al., 2019) (no fine tuning, default hyperparameters).

Fine-tuned BERT. We fine-tune the BERT model with a learning rate of 5e-5, and AdamW optimizer. We set the gradients to zero at every training batch. Then we use softmax activation to gate the output with the threshold of 0.5.

Fine-tuned T5. To fine-tune the text-to-text transformer (Raffel et al., 2020), we use T5ForConditionalGeneration approach (equally to question-answering task) where the input is a sentence (as a question), and the output is an answer (as a label). We use a learning rate of 3e-4, with max sequence length of 512.

Linear-Neuron Attention BERT. We replicate...
the winning approach of the NLP4IF’19 shared-task (Mapes et al., 2019). It relies on BERT architecture with some modifications of hyperparameters (sentence length of 50 tokens, a learning rate of 1e-5, along with 12 attention heads and 12 transformer blocks). It uses the linear neuron without an activation function, and a threshold of 0.3 for the final prediction.

Multi-granularity BERT. This model (Da San Martino et al., 2019) relies on BERT transformer with multi-granularity network on top that has multi-classifiers for different granularity levels of text (e.g., document, paragraph, sentence, word, subword, and character-level). We replicate this model with BertAdam optimizer and ReLU activation function.

Multi-granularity + Featured BERT. We integrate the proposed features (Section 4) into (Da San Martino et al., 2019), taking only the last layer of sentence-level granularity. We feed the proposed features to a BERT classifier to obtain logits which then aggregate with the last layer of sentence-level granularity to produce predictions.

BERT + Featured BiLSTM. We build a pre-trained BERT transformer architecture, and Bi-directional Long Short-Term Memory (BiLSTM) architecture on top of the BERT model to handle the transformer architecture with our propaganda features. Firstly, the BERT model is used with learning rate of 0.001, with AdamW optimizer. We use the output of BERT that represents the [CLS] token of each sentence to combine with propaganda features as our input to the second model, the BiLSTM. For the BiLSTM model, after we feed our inputs of both [CLS] tokens combined with propaganda features, we train our BiLSTM model with hidden size of 256. Our BiLSTM hidden states consist of the last hidden states, and the last cell state for the BiLSTM layers. We then apply relu gate function, with a linear dense, then use a dropout function of 0.1. At the last layer, we use another linear dense layer to output final logits, then we apply a sigmoid activation function as final outputs.

BERT + Featured Logistic Regression. We use pre-trained BERT transformer architecture to output [CLS] token, then use this output to stack with another prediction model, i.e., logistic regression. We build a linear classifier and feed it with propaganda features as a dense layer. We then combine these logits with [CLS] tokens as the input to logistic regression on top of BERT.

5.2 Results and Error Analysis

Table 2 reports on the results obtained for the SLC task (propaganda vs no propaganda). We run each experiment 5 times and report the macro-average of all metrics. Our proposed models achieve the highest F1-score of 0.72 using BERT + Featured Logistic Regression model (persuasion, sentiment, and message simplicity features), and the highest precision-score 0.80 using BERT + Featured BiLSTM model on NLP4IF’19 dataset, outperforming the state-of-the-art models. For SemEval’20-T11, we do not have the scores from the challenge (the binary task was not proposed), but we compare the obtained results with the replicated architectures of SOTA models. Our proposed architecture obtained the best F1-score using BERT+Featured Logistic Regression. Using semantic features alone perform slightly better than combining them with argumentation features.

Table 3 reports on some misclassified examples of our best model on NLP4IF’19 dataset. Some short sentences containing strong intention keywords (e.g., “hate”, “slave”) have been missclass-
Table 3: Examples of misclassified sentences by the BERT + featured logistic regression model (NLP4IF’19)

<table>
<thead>
<tr>
<th>False Positive</th>
<th>False Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>People who hate freedom will get unfettered access to the minds of 2 billion people.</td>
<td>The American people have a right to know, and those that engaged in this type of behavior do not have a right to hide.</td>
</tr>
<tr>
<td>You are a slave to white America.</td>
<td>Hitler was a very great man.</td>
</tr>
</tbody>
</table>

Table 3: Examples of misclassified sentences by the BERT + featured logistic regression model (NLP4IF’19)

sified as false positives. As for false negatives, the underlined fragments are labeled propaganda in gold standard, but have not been recognized as such by the classifier (mainly informative statements).

6 Fragment-level Classification

In this section, we address the task of fragment-level classification, meaning that both the spans and the type of propaganda technique should be identified in the sentences. Again, to test the proposed methods, we use both NLP4IF’19 and SemEval’20 T11 datasets. However, in the two challenges, the FLC task was evaluated according to different strategies, explained in the following.

6.1 Task 1: FLC on NLP4IF’19 Dataset

In the NLP4IF’19 dataset, 18 propaganda techniques are annotated. Prediction is expected to be at token-level. Multiple tokens can belong to the same span, and annotated with one propaganda type. Tokens that do not carry any propaganda bias are annotated as “no propaganda”. To perform tokenization we run the tokenizer provided with the pretrained model of each transformer.

6.1.1 Prediction Models

**Fine-tuned BERT (baseline).** Pretrained bert-base-uncased model and BERT architecture (Devlin et al., 2019) with default hyperparameters. Our implementation is based on huggingface transformers. Settings: learning rate of 5e-5, padded length of 128, and batch_size of 16. We use CrossEntropyLoss as a loss function, and softmax activation function to gate output neurons.

**Fine-tuned RoBERTa (baseline).** We use roberta-base model with the same hyperparameters of loss and activation functions as the fine-tuned BERT model mentioned above.

**State-of-the-art Model.** The winning team applied BERT architecture for token classification (Yoosuf and Yang, 2019) on 20 labels (i.e., 18 propaganda classes, plus “background” as non-propaganda, and “auxiliary” for fractions of previous tokens). They use a BERT language model, then apply softmax function, followed by a linear multi-label classification layer to output their predictions.

**Transformer + CRF.** We use a pre-trained model base-uncased with a learning rate of 3e-5 for BERT transformer, and a pre-trained model roberta-base with a learning rate of 2e-5 for RoBERTa transformers (hyperparameters: dropout of 0.1 with the max_length of 128, batch_size of 16 with AdamW optimizer and CrossEntropy loss function). We use CRF layer as the final layer.

6.1.2 Results and Error Analysis

Table 4 reports on the obtained performances. Evaluation is reported as the average of micro-F1 scores of 5 run-times (we use the evaluation scripts provided by (Da San Martino et al., 2019)).

The proposed architecture based on transformers with CRF output layer at different learning gradients (epochs) outperforms SOTA model on several propaganda techniques at different learning gradient ranging from 5 to 15 epochs. We also tested other architectures such as Transformer+CRF with less learning gradients (3 epochs), Transformer architecture with semantic and/or argumentation features + CRF layer by adding extracted features from sentence-level (Section 4) to each token of its sentence to a linear layer before a loss function, with no major improvements.

In Table 4, we compare the performances of the proposed models w.r.t. the SOTA (Yoosuf and Yang, 2019), on the most frequent classes. Table 5 reports examples of misclassification related to that technique. We observe that our proposed model does not capture well the articles (i.e., it, as, an, the), but rather focuses on capturing intentional word tokens (i.e., white, unbelievably, rude, wonderful, treasonous). As for future work to improve results on this specific category, we will investigate the work of (Habernal et al., 2018) according to which a dedicated strategy is needed.

6.2 Task 2: FLC on SemEval’20 T11 Dataset

In SemEval’20 T11 dataset, 14 propaganda techniques are annotated. We focus here on the task called Technique-Classification task (TC). We cast it as a sentence-span classification problem, where...
we combine logits of tokenized elements from the sentence and the span, to learn the prediction. Moreover, we add the semantic and argumentation features to enhance the performance.

As pre-processing, both the tokenized sentence and the span are used to feed the transformer (Huggingface tokenizers) as follows: \(i\) we input a sentence to the tokenizer where max_length is set to 128 with padding; \(ii\) we input the span provided by the propaganda span-template published by the workshop, and we set max_length value of 20 with padding. If a sentence does not contain propaganda spans, it is labeled as a “none-propaganda”.

### 6.2.1 Prediction Models

**Baseline.** For all the tested architectures (BERT and RoBERTa), we use the same type of transformer model to produce logits \(L\) regarding the sentence-level and span-level individually. For BERT model, we use pre-trained model bert-base-uncased, learning rate of 5e-5, and \(\alpha\) of 0.1. For RoBERTa, we take roberta-base pre-trained model with learning rate of 2e-5 with \(\alpha\) of 0.5. All transformer models apply Adam optimizer, dropout 0.1, and CrossEntropy as a loss function per sentence \(\text{loss}_{\text{sentence}}\) and span \(\text{loss}_{\text{span}}\).

We arrange these alignment of \(L\) to calculate the average loss as joint loss \(\text{loss}_{\text{joint}}\) from each loss element. Here we introduce a \(\text{loss}_{\text{joint}}\) function before back-propagation:

\[
\text{loss}_{\text{joint}} = \alpha \times \frac{\text{loss}_{\text{sentence}} + \text{loss}_{\text{span}}}{N_{\text{loss}}}
\]

where \(N_{\text{loss}}\) stands for a number of loss elements that are taken into the model.

**State-of-the-art Model.** The winning team (Jurkiewicz et al., 2020) applies RoBERTa (roberta-large) with pre-trained model. The training set is increased with silver annotation based on gold annotation, and then another RoBERTa model is stacked on top to output the predictions.

**Proposed Architecture.** We propose another set of elements to feed the transformer by introducing the semantic and argumentation features into BiLSTM layer to produce \(L\) of proposed features, then we apply CrossEntropy as a loss function of our BiLSTM as \(\text{loss}_{\text{proposed features}}\) then perform an addition with other loss in the \(\text{loss}_{\text{joint}}\) function as follows:

\[
\text{loss}_{\text{joint}} = \alpha \times \frac{\text{loss}_{\text{sentence}} + \text{loss}_{\text{span}} + \text{loss}_{\text{proposed features}}}{N_{\text{loss}}}
\]

Hyper-parameters: 256 hidden_size, 1 hidden_layer, drop_out of 0.1 with ReLU function at the last layer before the joint loss function.

### 6.2.2 Results and Error Analysis

As mentioned before, the gold labels of the test set of SemEval’20 T11 are not available, but it is possible to submit a system run to the challenge website and to obtain the evaluation score. The evaluation system only accepts the exact list...
of span-templates of the test set (partial overlapping spans or missing spans are not accepted). Table 6 reports on the obtained results (through such evaluation system) on 5 runs as micro-F1. Scores in bold are the ones for which significant improvement can be observed w.r.t. SOTA model. RoBERTa with argumentation features can outperform results on “Thought-terminating_Cliches”. Moreover, by using all semantic and argumentation features together, we can obtain some improvements over “Bandwagon, Reductio_ad_hilterum” and “Casual-Oversimplification”. Table 7 shows some examples of misclassified instances. In general, we noticed that using different training epochs help detecting different propaganda techniques. In particular, it is observed that some techniques tend to be learnt best at low training epochs (i.e., “Bandwagon, Reductio_ad_hilterum”, “Thought-terminating_Cliches”), some at high training epochs (i.e., “Casual-Oversimplification”).

### 7 Concluding Remarks

In this paper, we proposed a new neural architecture combined with state-of-the-art language models and a rich set of linguistic features for the detection of propaganda messages in text, and their further classification along with standard propaganda techniques. Despite the boost in accuracy we achieved on two standard benchmarks for propaganda detection and classification (≈ 10% of F1 scores on sentence-level classification and on specific propaganda techniques on fragment-level classification), this task remains challenging, in particular regarding the fine-grained classification of the different propaganda classes. The state-of-the-art results on this subtask require further improvement to actually embed these solutions in real-world systems.

Future work goes in this direction, with the aim to improve the performance both of the disinformation detection task and of the classification of propaganda techniques. Moreover, we are currently investigating the propaganda classes we discussed in this paper in the context of political debates, with the aim of building a fallacy detection system that relies on the identification of propagandist messages in political speeches.

### Acknowledgments

This work is partially supported by the ANSER project PIA FSN2 n. P159564-2661789/DOSS0060094 between Inria and Qwant. Moreover, this work has been supported by the French government, through the 3IA Côte d’Azur Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002.

### References


Harold Dwight Lasswell. 1938. *Propaganda technique in the world war.*


Abstract
The current natural language processing is strongly focused on raising accuracy. The progress comes at a cost of super-heavy models with hundreds of millions or even billions of parameters. However, simple syntactic tasks such as part-of-speech (POS) tagging, dependency parsing or named entity recognition (NER) do not require the largest models to achieve acceptable results. In line with this assumption we try to minimize the size of the model that jointly performs all three tasks. We introduce ComboNER: a lightweight tool, orders of magnitude smaller than state-of-the-art transformers. It is based on pre-trained subword embeddings and recurrent neural network architecture. ComboNER operates on Polish language data. The model has outputs for POS tagging, dependency parsing and NER. Our paper contains some insights from fine-tuning of the model and reports its overall results.

1 Introduction
The paradigm for automated language processing was originally based on building dedicated tools for every task. We illustrate it on the example of the Polish language, but it is very likely the case also for other languages. The dedicated tools include morphological taggers and disambiguators (a Polish language approximation of POS tagging), dedicated dependency parsers and named entity recognizers. For convenience, the tools sometimes were made available as web services and possibly composed into chains. One such example is Multiservice (Ogrodniczuk and Lenart, 2012), a linguistic Web service for Polish, combining several mature offline linguistic tools in a common online platform. Multiservice offers a number of pre-defined chains. For example, the dependency parser requires morphological tagging and disambiguation as the initial step. The two tools are put together in appropriate order in one chain.

Among universal tools (not dedicated to the Polish language) it is worth to mention:

• UDPipe, a pipeline (tool) for tokenization, tagging, lemmatization and dependency parsing (Straka et al., 2016).

• NLP-Cube which in addition to the capabilities mentioned for UDPipe offers sentence segmentation and lemmatization. The reported results cover many languages but not Polish (Boros et al., 2018).

• Stanza, a tool for tokenization, multi-word token (MWT) expansion, lemmatization, part-of-speech (POS) and morphological features tagging, dependency parsing, and named entity recognition. Named entity recognition models are separate and Polish is not available (Qi et al., 2020).

Recently, with the advances of neural networks in natural language processing, new tools were proposed that simplify the chains by doing several tasks at once. One example is Combo (Rybak and Wróblewska, 2018), a bi-directional recurrent neural network (RNN) using word2vec embeddings with multiple outputs for morphological and dependency analysis. The model was submitted to the CoNLL 2018 Universal Dependency shared task.

The Combo tool, while doing certain things right, has its own issues. The biggest problem is the fact that it requires word-level word2vec embeddings for every token that it encounters. Unfortunately, the number of unique tokens in the Polish language is rather large. For example, in the National Corpus of Polish and Wikipedia, the number of unique case-sensitive tokens exceeds 2 millions. This is by far too many to fit into GPU memory. This
leads to a situation where only word2vec vectors that are used in input texts are loaded into GPU memory (Embedding layer) and an additional preprocessing step of data-driven word vector selection and loading. Depending on linguistic productivity, this operation has to be repeated once in a while when processing large amounts of text. To solve this problem, one can apply techniques of vocabulary selection such as those described in (Chen et al., 2019). Hopefully, another simple solution exists for this issue: subword level embeddings (Sennrich et al., 2016). This idea, successfully applied in machine translation, became popular in multiple other scenarios and architectures, including transformer neural networks such as BERT (Devlin et al., 2019). It allows to represent an unknown word (not present in the dictionary) as a sequence of shorter tokens or subwords. Using subword-level tokenization and embeddings the problem of vocabulary selection is non-existent.

The goal of ComboNER is to (1) avoid the above issues of the initial Combo (Rybak and Wróblewska, 2018) by introducing subword tokenization and (2) add named entity recognition as another output. The overall design goal is to reduce the size of the model where possible.

2 Datasets

This section describes the datasets used to train the ComboNER model. It is only an overview with basic information about data sizes, please refer the cited papers for more details such as label descriptions and annotation principles.

2.1 POS and Dependency

We trained the POS and dependency outputs of the ComboNER model using the Polish language subset of the Universal Dependencies (UD) treebank (Nivre et al., 2020). It contains 17723 sentences in the train set and 2215 sentences in the test and dev sets. Sentences are annotated for part-of-speech, morphological information and dependencies (heads and labels).

2.2 Named Entity

To train the named entity output of ComboNER, we used the relevant subset of the National Corpus of Polish (Przepiorkowski et al., 2012). For label descriptions and annotation principles for named entities refer to the chapter 9.

Overall, the dataset contains 39534 texts and 85816 sentences. Since ComboNER requires exactly one sentence as its input (a limit imposed by the dependency structure defined for a single sentence only), we splitted the named entity data into single sentences using the Polish language model of the spacy.io. We then randomly divided the data into train and test part. This resulted in 81470 sentences (95%) in the train set and 4346 (5%) in the test set.

3 Hyperparameter Tuning

To tune the parameters we used only POS accuracy and one measure from the dependency parsing: UAS score (unlabeled attachment score). This was in order to simplify the analysis and check the balance between types of outputs. Therefore, the hyperparameter tuning was carried out only on the Universal Dependencies dataset.

Hyper-parameter tuning was done using the HParams API in TensorFlow and grid-search approach.

Tuning was subjected to the following hyperparameters:

- Length of the subword embedding vector (emb_dim). Values tested: 25, 100, 200.
- The number of subwords for which we have vectors, i.e. the size of the vocabulary (vocab): 10,000 (10k), 100,000 (100k).
- The size of the Bi-LSTM cell output assuming a single layer model (rnn). Tested values: 100, 200.

ComboNER uses pre-trained Polish language subword embedding vectors from the BPEmb library (Heinzerling and Strube, 2018). Therefore, vocabulary and vector sizes are determined by availability in BPEmb.

For initial parameter selection only the results of the first training epoch were taken into account.

The Table 1 shows tuning results of the selected output as the accuracy of the POS tagger and dependency parser - where the UAS measure was selected. It also shows the mean of both measures in the AVG column.

1https://github.com/ipipan/spacy-pl
2We did not use a validation split because hyperparameter selection was focused on UD part of the data.
3https://www.tensorflow.org/tensorboard/hyperparameter_tuning_with_hparams
The largest embeddings (emb_dim equal to 200) work best in combination with size-corresponding Bi-LSTM cells (rnn equal to 200). This observation concerns both the accuracy of POS tagging and the UAS measure of the dependency parser.

Interestingly, the number of subwords, i.e. vocabulary size (vocab), turned out to be a non-obvious measure. The POS tagger performed better with a large number of subwords (100k vocab), while a dependency parser preferred a smaller number (10k vocab). This effect was present regardless of RNN cell size.

We also tested the configuration of the model with two Bi-LSTM layers. We tested two values of LSTM cell sizes for the first layer (100, 200) and two (25, 50) for the second layer. When measuring performance for the first training epoch, a model with a two-layer Bi-LSTM performs generally worse than a single-layer model with similar settings. As a result, the addition of a second Bi-LSTM layer does not seem to be advisable.

For the named entity recognition, two variants of the model architecture were tried. In the Multiservice (Ogrodniczuk and Lenart, 2012) and in many older generation NLP tools, named entity recognition model uses POS tags as an input feature and is therefore located after POS tagging. This observation leads to testing two design choices for the named entity recognition part of ComboNER. The first one is two independent outputs for NER and POS / dependency: the outputs share only the word embedding layer. The second assumes dependence: named entity branch takes as an input not only embeddings but also hidden states from the LSTM in POS / dependency branch. This setting allows named entity recognition to take into account information from layers trained on syntactic data, in manner somewhat similar to using POS features in older NLP models.

The results in terms of F1 scores of both variants are presented in Table 2. They clearly indicate that syntactic information helps in named entity recognition as the overall gain computed as an average for all entity categories is 12.8%.

<table>
<thead>
<tr>
<th>Dep.</th>
<th>Indep.</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>0.797</td>
<td>0.702</td>
</tr>
<tr>
<td>OrgName</td>
<td>0.427</td>
<td>0.213</td>
</tr>
<tr>
<td>PersName</td>
<td>0.683</td>
<td>0.634</td>
</tr>
<tr>
<td>PlaceName</td>
<td>0.564</td>
<td>0.410</td>
</tr>
<tr>
<td>average</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: F1 score of two variants of the NER output: (Inep.) independent outputs for NER and POS / dependency parsing, (Dep.) named entity branch takes as an input not only embeddings but also hidden states from the LSTM in POS / dependency.

4 ComboNER Architecture

Figure 1 illustrates the design of ComboNER. Data processing starts with subword tokenization and embedding layer. Here, pre-trained subword
embedding vectors are used (along with the tokenizer), namely monolingual Polish language BPEmb (Heinzerling and Strube, 2018) trained on Wikipedia. This part is common to all outputs. Then, computation is split into two branches: one for POS and dependency, the other for named entities. Each of the two branches begins with a bi-directional LSTM layer. Named entity branch takes as input not only embeddings but also hidden states from the LSTM in POS / dependency branch. Architecture of this type was selected on the basis of experiments described in Table 2. Intuitively, part-of-speech information is often useful for named entity recognition; in certain natural language processing pipelines such as Multiservice (Ogrodniczuk and Lenart, 2012) named entity recognition (NER) uses part-of-speech (POS) features and therefore is executed downstream (after) POS.

Activations inside the network are tanh and softmax is used in the final layers. For details of implementation see (Rybak and Wróblewska, 2018); in particular, dependency parser outputs with the dot product solution follow closely the initial model.

5 Final Model

5.1 Model Parameters

For the final model, we selected following hyper-parameters:

- 100 output units in the POS and dependency LSTM,
- 50 output units in the named entity LSTM,
- dropout rate of 0.2,
- learning rate of 3e-4,
- 100 output units of the Dense layer for POS output,
- 100 output units of the Dense layer for dependency output,
- vocabulary size of 50,000 (50k),
- embedding size of 100.

The selection of embedding size 100 and 50k vocabulary is a compromise between the quality of POS and dependency outputs.

The number of parameters of the ComboNER is 5.3 millions. The size of the embedding layer alone is 5 millions (embedding size times vocabulary) and we make it trainable for both training tasks as it improves the overall results marginally. The remaining parts of the model are rather modest with named entity part contributing over 100k parameters. The disk size of the model slightly exceeds 200 MB.

The model handles sentences up to 67 subword tokens long as this was the maximum sentence length encountered in the UD dataset. It assumes that the input consists of a single sentence. No
special tokens are needed to indicate beginning or end of sentence.

5.2 Evaluations

We trained the model 45 epochs on the Polish language UD dataset (POS and dependency part) and 20 epochs on the subset NKJP (the National Corpus of Polish, named entity labels) as described in Section 2. The execution time of a single training epoch was 3 minutes on the UD treebank (POS and dependency) and about 10 minutes on the named entity data on a Tesla V100-SXM2-16GB GPU card.

Table 3 contains the POS results of the final fully trained output of ComboNER. The overall accuracy (not reported in the Table) is 0.933. The performance is satisfactory for most parts-of-speech. The worst performing one is PROPN (proper names), due to high lexical variation and relatively low frequency in the UD corpus.

In the case of dependency parser outputs, the two relevant measures are unlabelled attachment score (UAS) and labelled attachment score (LAS). The values measured are 0.71 for UAS and 0.858 for LAS score.

Table 4 contains the results of named entity evaluation. Two categories, present in the data, did not provide reasonable results: GeogName and Time. This is due to low counts, as these categories were the least frequent in the dataset. In the case of categories: Date, OrgName, PersName and PlaceName ComboNER performed with reasonable precision and recall.

6 Conclusions

This paper describes ComboNER: a new version of the Combo tool (Rybak and Wróblewska, 2018), developed with following assumptions in mind: (1) use subwords to avoid out-of-vocabulary issues (words in model’s input not present in memory / the embedding layer), (2) add named entity recognition output, (3) lightweight in terms of memory footprint, as of today’s standards. The tool was implemented in TensorFlow 2.

ComboNER follows many design choices of the original Combo (Rybak and Wróblewska, 2018), but introduces some new solutions.

Training multi-output models on multiple datasets is a challenging task. Technically, it was solved by freezing appropriate part (layers) of the model while training another part, using a dedicated TensorFlow data loader class for each corpus.

The paper contains the results of hyperparameter optimization, which dictate the final hyperparameter choice for the ComboNER model. There are several interesting findings. First, the number of subwords (vocabulary size): the POS tagger performed better with a large number of subwords (100k vocab), while dependency parser preferred a smaller number (10k vocab). Second, named entity layers greatly benefit from accessing layers pre-trained for POS and dependency. This architecture improves the F1 score of named entity recognition by 12.8% on average compared to the variant with fully independent training of both tasks.

The results of PolEval competition in 2018 are an interesting reference for comparisons regarding dependency parser and named entity recognition quality (Ogrodniczuk and Kobyliński, 2018)\(^4\). It appears that LAS score achieved by the ComboNER is competitive and easily on-par with participating systems. Unfortunately, the solutions for named entity recognition outperform the output of ComboNER in a significant manner. One explanation of this fact is context: effective named entity recognition requires context wider than just the input sentence. Most of the corpora and tools assume the supra-sentence level. Important clues for identifying named entities are often a part of preceding sentences.

In order to evaluate the POS tagger, one can compare the results of the PolEval competition in 2017(Kobyliński and Ogrodniczuk, 2017)\(^5\). In this context, the accuracy of 0.933 is very competitive and on-par with the best of the participating systems.

To recap, this paper describes a relatively straightforward yet important modifications over previous work (Rybak and Wróblewska, 2018). Our goals were to develop a small system and avoid large scale transformer, at the same time solving out-of-vocabulary problems posed by word-level embeddings. In the future it may be advisable to compare with small transformer models like DistilBERT (Sanh et al., 2020) (40% less parameters than the BERT base model).

\(^4\)http://2018.poleval.pl/index.php/results/
\(^5\)http://2017.poleval.pl/index.php/results/
<table>
<thead>
<tr>
<th></th>
<th>prec</th>
<th>recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>0.983</td>
<td>0.968</td>
<td>0.975</td>
</tr>
<tr>
<td>ADP</td>
<td>0.935</td>
<td>0.979</td>
<td>0.957</td>
</tr>
<tr>
<td>ADV</td>
<td>0.788</td>
<td>0.630</td>
<td>0.700</td>
</tr>
<tr>
<td>AUX</td>
<td>0.812</td>
<td>0.904</td>
<td>0.856</td>
</tr>
<tr>
<td>CCONJ</td>
<td>0.929</td>
<td>0.892</td>
<td>0.910</td>
</tr>
<tr>
<td>DET</td>
<td>0.882</td>
<td>0.772</td>
<td>0.823</td>
</tr>
<tr>
<td>NOUN</td>
<td>0.750</td>
<td>0.859</td>
<td>0.801</td>
</tr>
<tr>
<td>NUM</td>
<td>0.783</td>
<td>0.757</td>
<td>0.770</td>
</tr>
<tr>
<td>PART</td>
<td>0.857</td>
<td>0.742</td>
<td>0.795</td>
</tr>
<tr>
<td>PRON</td>
<td>0.905</td>
<td>0.948</td>
<td>0.926</td>
</tr>
<tr>
<td>PROPN</td>
<td>0.754</td>
<td>0.602</td>
<td>0.670</td>
</tr>
<tr>
<td>PUNCT</td>
<td>0.997</td>
<td>9.996</td>
<td>0.997</td>
</tr>
<tr>
<td>SCONJ</td>
<td>0.876</td>
<td>0.911</td>
<td>0.893</td>
</tr>
<tr>
<td>VERB</td>
<td>0.876</td>
<td>0.911</td>
<td>0.893</td>
</tr>
</tbody>
</table>

Table 3: Part-of-speech evaluation of the fully trained model. Precision (prec), recall and F1 measured on the test set.

<table>
<thead>
<tr>
<th></th>
<th>prec</th>
<th>recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>0.891</td>
<td>0.721</td>
<td>0.797</td>
</tr>
<tr>
<td>OrgName</td>
<td>0.640</td>
<td>0.320</td>
<td>0.427</td>
</tr>
<tr>
<td>PersName</td>
<td>0.747</td>
<td>0.629</td>
<td>0.683</td>
</tr>
<tr>
<td>PlaceName</td>
<td>0.653</td>
<td>0.496</td>
<td>0.564</td>
</tr>
</tbody>
</table>

Table 4: Named entity recognition evaluation of the fully trained model. Precision (prec), recall and F1 measured on the test set. GeogName was skipped due to low results.

6.1 Software and Data
Source codes and serialized models are available at https://github.com/CLARIN-PL/ComboNER/ and https://github.com/alexwz/ComboNER.

Acknowledgments
Work financed by the Polish Ministry of Science and Higher Education, a program in support of scientific units involved in the development of a European research infrastructure for the humanities and social sciences in the scope of the consortium CLARIN ERIC 2018-2021.

References


Investigating Annotator Bias in Abusive Language Datasets

Maximilian Wich, Christian Widmer, Gerhard Hagerer, Georg Groh
Technical University of Munich, Germany
{maximilian.wich, christian.widmer, gerhard.hagerer}@tum.de, grohg@in.tum.de

Abstract

Nowadays, social media platforms use classification models to cope with hate speech and abusive language. The problem of these models is their vulnerability to bias. A prevalent form of bias in hate speech and abusive language datasets is annotator bias caused by the annotators subjective perception and the complexity of the annotation task. In our paper, we develop a set of methods to measure annotator bias in abusive language datasets and to identify different perspectives on abusive language. We apply these methods to four different abusive language datasets. Our proposed approach supports annotation processes of such datasets and future research addressing different perspectives on the perception of abusive language.

1 Introduction

A challenge that social media platforms are facing in recent years is the large amount of hate speech and other forms of abusive language (Duggan, 2017). Manual monitoring, however, is no longer possible due to the vast volume of user-generated content. Therefore, machine learning models are trained and used by social media platforms, such as Facebook, to automatically detect such content (Kantor, 2020). According to Rose (2021), these models are a key component of Facebook’s fight against hate speech.

A problem with such machine learning models is that they are vulnerable to bias (Vidgen and Derczynski, 2021; Dixon et al., 2018). Biased models can strongly impair the fairness of a system, which can lead to discrimination (Dixon et al., 2018).

Bias in abusive language detection is already a topic that researchers have started to investigate (Vidgen and Derczynski, 2021; Wich et al., 2021). The type of bias we will focus on in this study is annotator bias. This form of bias is a result of annotators who perceive abusive language differently from each other and have different levels of experience and knowledge (Ross et al., 2016; Waseem, 2016; Geva et al., 2019; Wich et al., 2020).

We aim to investigate two aspects of annotator bias. (1) Assuming that there is only one perspective (one truth) on whether a text is abusive or not, we develop an approach to measure and visualize annotator bias. This approach optimizes the annotation process (e.g., outlier detection, adapting appropriate annotation guidelines). (2) Acknowledging multiple valid views on a text (e.g., a group has a more liberal attitude towards abusive texts, while another is stricter), we aim to identify annotator groups to model different, yet valid perspectives. The questions resulting from these research objectives are the following:

- **RQ1**: How can we measure and visualize annotator bias in abusive language datasets?

- **RQ2**: How can we identify and visualize different annotator perspectives on abusive language?

Our contributions are the following:

1. To characterize annotators, we gauge how liberal or strict they annotate in comparison to other annotators. To model annotator bias, we calculate a pessimistic and optimistic score for each annotator that can be visualized in different ways (e.g., scatter plot, cluster map).

2. To identify annotator groups with different perspectives on abusive language, we utilize a classifier-based method with the proposed approach, which is applied to one dataset.
2 Related Work

Hate speech and abusive language detection have gained a lot of attention in recent years. A range of different datasets (Vidgen and Derczynski, 2021) and shared tasks (Basile et al., 2019; Zampieri et al., 2019, 2020) were published to foster research in this area. Most of the datasets are commonly labeled by crowdworkers or those in academia with varying expertise (Vidgen and Derczynski, 2021). However, human annotations tend to be subjective and thus inconsistent (Aroyo and Welty, 2015), at least if not moderated very strictly. Especially for abusive language, Salminen et al. (2018) show that individuals interpret hate speech differently. One common method to improve the label quality is presenting each sample to multiple annotators and aggregate their results (Sheng et al., 2008). Dawid and Skene (1979) were the first to propose an approach that incorporates annotator quality into label aggregation. Their expectation-maximization (EM) algorithm uses the bias matrices to estimate the latent truth. In the matrices the annotator quality is encoded. Their seminal work led to further improvements and methods (Whitehill et al., 2009; Raykar and Yu, 2012; Hovy et al., 2013). For NLP tasks, Snow et al. (2008) used a customized Dawid-Skene algorithm to correct for individual biases of crowdworkers and improve model accuracy. However, they did not quantify and inspect the bias of the annotators.

In abusive language detection, annotator bias research has focused on how the annotators background influences their annotations. Waseem (2016) found models trained on crowd annotations are outperformed by models trained on expert annotations. Ross et al. (2016) emphasized the importance of detailed guidelines to achieve reliable annotations. Binns et al. (2017) showed that classifiers trained on annotations differ in their performance on test data annotated by men or women. Al Kuwatly et al. (2020) picked up this approach, enhanced it, included other demographics, and found significant differences for annotator’s age group and educational background. Sap et al. (2019) observed that posts in African American dialect are more likely to be labeled offensive. Similarly, Larimore et al. (2021) found that white and non-white workers annotate racially sensitive topics differently. Apart from studying the demographic background, researchers also attempt to find groups of annotators with common annotation behavior. Wich et al. (2020) use graph methods to cluster annotators in groups with higher inter-annotator agreement within groups than across groups. Akhtar et al. (2020) defined a polarization measure to split annotators in two groups that maximize opposing annotations. To the best of our knowledge, no one has quantified annotator bias at the annotator level. Furthermore, the hypothesis of multiple perspectives on abusive language is rarely investigated.

3 Datasets

We use four different abusive English language datasets to demonstrate our proposed approach. It was challenging to find appropriate datasets because our experiment requires unaggregated annotation data. Most of the abusive language datasets contain only the agreed upon labels in the documentation and not the individual votes of the annotators.

Table 1 lists the four datasets with additional in-
Figure 1: Box plots of the annotators’ inter-rater reliability scores.

formation. The first three datasets (Vidgen, Guest, and Kurrek) are similar because they are annotated by small groups of annotators (between 6 and 26). Furthermore, each document of the three datasets was annotated by two annotators. In case of disagreement, an expert reviewed the votes and decided on the gold label. In contrast, the Wulczyn dataset was annotated by many crowdworkers—a typical crowdsourcing setup: a group of workers who annotated a small number of documents. Each document was annotated by up to 10 annotators. In case of ambiguous annotations, an expert review did not take place. The gold label was determined based on majority vote. For our experiment, we convert all datasets to a binary task (abusive/neutral) to compare the results.

Figure 1 shows the distributions of the annotators’ inter-rater reliability scores in form of Krippendorff’s alpha. The colored dots represent the overall inter-rater reliability score of each dataset. We see that the overall Krippendorff’s alphas are all in the same range. The Wulczyn dataset was annotated by many crowdworkers—a typical crowdsourcing setup: a group of workers who annotated a small number of documents. Each document was annotated by up to 10 annotators. In case of ambiguous annotations, an expert review did not take place. The gold label was determined based on majority vote. For our experiment, we convert all datasets to a binary task (abusive/neutral) to compare the results.

4 Methodology

Our analysis of the annotator bias in the selected abusive language datasets consists of two parts. In the first part, we characterize the annotation behavior based on the deviations of the annotator votes compared with the gold standard of the dataset. In the second part, we visualize the perspectives of different annotator groups on abusive language with the aid of classification models.

4.1 Characterizing Annotator Bias

We define annotator bias as the deviations between the annotator votes and the gold labels of the dataset. The gold labels are either the final labels of the dataset or majority of the single votes. To measure the annotator bias, we use the concept of the confusion matrix. Figure 2 shows a matrix for a binary classification task of abusive documents (neutral/abusive). The rows represent the classes of the gold labels; the columns represent the classes observed by the annotator. The bias matrix quantifies the deviations between the labels observed by the annotator and the gold labels. Each annotator has one bias matrix.

We use cells II and III, which represent false negatives (type II error) and false positives (type I error) in the classical confusion matrix, to characterize the annotators’ behavior, and we assign each annotator a pessimistic and optimistic score. Cell II reflects the number of documents that are neutral according to the gold standard but that are annotated as abusive by the annotator, signaling that the annotator is pessimistic in these cases. Cell III is the opposite, and shows the number of documents that are labeled as abusive according to the gold standard but perceived as neutral by the annotator, signaling that the annotator is optimistic in these cases. The pessimistic ($p_i$) and optimistic ($o_i$) scores of an annotator ($i$) are entries II and III of row-normalized bias matrix. The concept of annotator’s optimism and pessimism was proposed by Dawid and Skene (1979). This method also works if we have more than two classes as long as they are ordinal. In this case, the cells above or below the diagonal are summed up. In our paper, however, we consider only binary classification of tasks.

To analyze bias matrices, we utilize these options:
1. We calculate the bias matrix for a group of annotators or all of them by averaging the bias matrices. The resulting bias matrix delineates whether the selected annotators tend to be optimistic or pessimistic. These findings assist with adjustment of annotation guidelines or the training of the annotators.

2. We utilize a 2-dimensional scatter plot of the pessimistic and optimistic scores to visualize the annotators and their biases. In contrast to comparison of inter-rater reliability scores, this visualization reveals whether annotators are more optimistic or pessimistic than the gold standard. Such information can help to detect outliers in the respective direction and to instruct the identified annotators as appropriate.

3. We can calculate a distance between two annotators based on their bias matrices ($A$ and $B$) with the Frobenius norm (Golub and Van Loan, 2013, p.71):

$$
\text{distance}(A, B) = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} (a_{ij} - b_{ij})^2}
$$

Visualizing these distances with a hierarchically clustered heatmap helps identify annotator groups with similar annotation behavior and outliers.

4. If the number of annotators is so large that the results of the previously proposed methods is no longer manageable, we can apply a hierarchical clustering on the bias matrices based on our distance metric. By doing so, annotators with a similar annotation behavior are clustered. If we aggregate the bias matrices according to (1), we observe how the cluster annotated the data in context of the gold standard.

5. If we have additional information about the annotators to characterize them (e.g., demographics, such as age or education), we can use the pessimistic and optimistic scores to test whether there is a significant difference between the annotation behavior of annotators with different characteristics. For this purpose, we apply the two-dimensional Kolmogorov-Smirnov (KS2D) test (Fasano and Franceschini, 1987; Peacock, 1983) to compare the distributions of the groups’ pessimistic and optimistic scores. The output of the test is the Kolmogorov-Smirnov statistic $D$ and the corresponding significance level $s$. If $D$ is larger than the predefined significance level $p$ and $p$ is larger than $s$, we can reject the null hypothesis that both samples have the same distribution. We use the Python implementation provided by Gabriel Taillon\textsuperscript{1}. In the case of the Wulczyn dataset, we have such information (Wulczyn et al., 2017). Our predefined significance level $p$ is 0.05.

4.2 Identifying Different Perspectives on Abusive Language

The previous subsection focuses on methods to measure and visualize annotator bias, answering RQ1. The underlying assumption is that there is one truth, meaning one valid perspective on abusive language, and we want to identify outliers deviating from the one truth.

Now we assume that there are more perspectives on abusive language—e.g., a group has a more liberal attitude toward abusive texts, while another group is less liberal. To examine this hypothesis, we run the following experiment. First, we split the annotators into different groups based on the pessimistic and optimistic scores. Second, for each group we create a dataset, containing the documents that all groups annotated. The labels of the documents result from the majority vote of the groups’ annotators. Third, for each group we train a classification model on its training set and evaluate it on the test sets of all groups. Suppose a classifier performs well on its test set and worse on the other test sets. Thus, the performance is comparable to a baseline classifier trained on the same data with gold labels. In that case, it indicates that this group has a coherent perspective on abusive language that differs from the other groups. This approach is based on the method proposed by Wich et al. (2020).

To split the annotators according to their pessimistic ($p_i$) and optimistic $o_i$ scores, we apply the following function:

$$
\text{group}_i(p_i, o_i) = \begin{cases} 
0 & \text{if } p_i \geq 3 \cdot o_i \\
1 & \text{if } o_i \geq 3 \cdot p_i \\
2 & \text{otherwise} 
\end{cases}
$$

The factor 3 in the function is the result of a trade-off between having a dominating dimension

---

\textsuperscript{1}https://github.com/Gabinou/2DKS
in the optimistic and pessimistic group and having enough annotators in all three groups. Increasing the factor would strengthen the dominating dimensions but reduce the number of annotators in the optimistic and pessimistic groups. Decreasing the factor would weaken the dominating dimension but increase the number of annotators in the groups.

For the classification model, we use the pre-trained English DistilBERT model 

\texttt{distilbert-base-uncased} provided by the Transformer Library from Hugging Face (Wolf et al., 2020); it is a more concise version of BERT (Sanh et al., 2019; Devlin et al., 2019) and provides a performance comparable to BERT for abusive language detection (Devlin et al., 2019).

We train each model for three epochs with a learning rate of $5 \cdot 10^{-5}$ and a batch size of 32. After the three epochs, we select the model with the lowest validation loss. 60% of the documents annotated by all groups are used as a training set, 20% as a validation set, and 20% as a test set. To compare the different classifiers, we use the macro F1 score.

5 Results

5.1 Characterizing Annotator Bias

Aggregated Bias Matrix

The problem of the inter-rater reliability analysis is that it does not reveal whether the annotators annotated more pessimistically or optimistically. This gap is addressed by the aggregated bias matrices, shown in Figure 3. The annotators of the datasets Vidgen, Guest, and Wulczyn tended to annotate more liberally because the optimistic scores (bottom-left cell) outweigh the pessimistic scores (upper-right cell). On the contrary, the annotators of the Kurrek dataset were stricter because 16% of non-derogatory documents were labeled as derogatory (pessimistic score), while only 4.5% (optimistic score) of the derogatory documents were labeled as non-derogatory.

Scatter Plot of Annotators

To gain a better understanding of the individual annotation behavior, we analyze the annotators based on their pessimistic and optimistic scores, shown in Figure 4. Considering the plots of Vidgen, Guest, and Kurrek, we observe that the annotators of Vidgen and Guest annotated more liberally due to the higher optimistic scores, while it is the opposite for the Kurrek dataset. Comparing the Guest dataset with the other two, we see that the annotators are less widely spread, meaning the annotation behavior is more coherent. Concerning the previously mentioned outliers of Vidgen and Kurrek, we can use the plots to better understand how they deviate. The outlier of Vidgen is the most right data point, the outlier of Kurrek is the uppermost data point. Their positions reveal that the outlier of Vidgen annotated more liberally, while the outlier of Kurrek was stricter. These findings can help instructors to guide the annotators if the method is used during the annotation process. A further observation concerning both datasets is that the density of annotators increases toward the origin of both dimensions. This indicates that most of the annotators have a similar annotation behavior.

In the case of the Wulczyn dataset, plotting each annotator as a data point would be confusing because the dataset contains 4,053 annotators. Therefore, we decided to cluster the annotators with a hierarchical clustering approach, facilitating data interpretation. We chose the agglomerative clustering approach with $k = 30$ and Euclidean distance function. The reason for $k = 30$ is that it is a manageable amount of data points on the scatter plot and it has the same order of magnitude as Vidgen and Kurrek. Figure 4d shows the annotators’ clusters. We observe the tendency of the annotators to annotate more liberally, as shown by the aggregated bias matrix in Figure 3d.
Figure 4: Annotators visualized based on their pessimistic and optimistic scores; in case of Wulczyn, annotators are hierarchically clustered.

**Cluster Map of Distances between Annotators**

A method to identify groups of annotators with similar annotation behavior is the hierarchically clustered heatmap based on the distances between the bias matrices of the annotators. Figure 5 shows the cluster map of the Kurrek dataset. The first thing that catches the reader’s eye is the first column and row. It shows the outlier of the dataset. Furthermore, we observe that the annotators Ann7, Ann13, Ann15, Ann3, and Ann5 (last five columns and rows) form a group. In Figure 4c, these annotators are the points above a pessimistic score of 0.2 and below 0.6. The other 15 annotators exhibit a more coherent annotation behavior. Due to the page restriction, we do not include the analysis for the other three datasets.

**Different Annotation Behavior of Demographic Groups**

The Wulczyn dataset contains demographic information for 2,190 of the 4,053 annotators (i.e., gender, age group, education, and English as the first language). We tested for each demographic feature whether there is a difference between the groups regarding the annotators’ pessimistic and optimistic scores. The result of the two-dimensional Kolmogorov-Smirnov test for the demographic feature of gender is the following:

\[
D_{\text{gender}} = 0.092 \quad s_{\text{gender}} = 0.005
\]

Based on this result, we can reject the null hypothesis \((p = 0.05)\). Consequently, there is a significant difference between the pessimistic and optimistic
scores of male and female annotators. Females are more pessimistic than males ($p_{female} = 0.107$ and $p_{male} = 0.090$), while the optimistic scores are comparable ($o_{female} = 0.192$ and $o_{male} = 0.199$). For the feature describing whether English is the first language of the annotator or not, we can also reject the null hypothesis:

$$D_{1st\ language} = 0.192 \quad s_{1st\ language} = 8.9 \times 10^{-8}$$

Native English speakers have a larger pessimistic score ($p_{native} = 0.093$ and $p_{non-native} = 0.117$) and a lower optimistic score than non-native speakers ($o_{native} = 0.160$ and $o_{non-native} = 0.204$).

Table 3a shows the output of the two-dimensional Kolmogorov-Smirnov test for the different age groups. We observe that there are significant differences in the distributions of the annotators’ pessimistic and optimistic scores between the age groups—except between the ages 30-45 and over 60 and 45-60 and over 60. Interestingly, the largest difference is between the age groups 18-30 and 45-60. While annotators between 45 and 60 are more pessimistic ($p_{45-60} = 0.146$ and $o_{45-60} = 0.128$), it is the opposite for annotators between 18 and 30 ($p_{18-30} = 0.08518$ and $o_{18-30} = 0.234$).

Table 3b shows the output for the different educational backgrounds. In contrast to the age groups, the scores of the annotators do not differ greatly between the groups; however, the difference between annotators who have a Bachelors and Masters degree is significant.

5.2 Identifying Different Perspectives on Abusive Language

Since this experiment requires a dataset with a large number of documents and annotators, we could conduct it only with the Wulczyn dataset. In the case of the other three datasets, the number of annotators is too small to meaningfully split the annotators into subsets and to have enough documents that were annotated by all subsets.

The results of the experiment to identify different perspectives and to answer RQ2 can be found in Table 2. It shows the different F1 scores for the abusive class of the classifiers that were trained on subsets of annotators (rows) and were evaluated on the test sets of these subgroups (columns).

<table>
<thead>
<tr>
<th></th>
<th>Pessimistic</th>
<th>Medium</th>
<th>Optimistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pessimistic</td>
<td>80.2</td>
<td>80.6</td>
<td>71.0</td>
</tr>
<tr>
<td>Medium</td>
<td>73.5</td>
<td>81.9</td>
<td>83.1</td>
</tr>
<tr>
<td>Optimistic</td>
<td>64.3</td>
<td>74.4</td>
<td>87.5</td>
</tr>
</tbody>
</table>

Table 2: F1 scores from classifiers of the different annotator subsets.

To answer our RQ2 on how to identify and visualize different perspectives on abusive language of the annotators, we need to focus on the pessimistic and optimistic data. We observe that the classifier trained on the annotations of the optimistic annotators performs best on its own test set (87.5%) and worst on the pessimistic test set (64.5%). When the classifier trained on the more pessimistic annotations, the result is the opposite. It performs most poorly on the optimistic test set (71.0%) and comparable well on its own test set (80.2%). Only on the test set of the medium group, the pessimistic classifier performs slightly better.

It is more relevant to our research question that the pessimistic and optimistic classifiers work well on their own test set but worse on the test set of the other extreme. The first fact indicates that the annotations are coherent, so that the classifier can learn patterns to identify abusive language. The second aspect shows that the labels of the pessimistic and optimistic subgroups’ dataset are so different that it can cause a difference of 9.2 or 23.2pp in the F1 score. Consequently, we conclude that the annotators of the pessimistic and optimistic subgroup have two different perspectives on abusive language.

An explanation for the more coherent results of the optimistic classifier can be the larger number
Table 3: Results of 2-dimensional Kolmogorov-Smirnov test for split according to demographic features and corresponding pessimistic and optimistic scores (Wulczyn dataset); first number in cells is $D$, second $s$; bold means rejected.

(a) Age group of annotators

<table>
<thead>
<tr>
<th>Age Group</th>
<th>18-30</th>
<th>30-45</th>
<th>45-60</th>
<th>Pessimistic</th>
<th>Optimistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Under 18</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.080</td>
<td>0.172</td>
</tr>
<tr>
<td>18-30</td>
<td>0.261 / 0.040</td>
<td>-</td>
<td>-</td>
<td>0.085</td>
<td>0.234</td>
</tr>
<tr>
<td>30-45</td>
<td>0.303 / 0.011</td>
<td>0.177 / 0.000</td>
<td>-</td>
<td>0.100</td>
<td>0.165</td>
</tr>
<tr>
<td>45-60</td>
<td>0.435 / 0.000</td>
<td>0.322 / 0.000</td>
<td>0.216 / 0.000</td>
<td>0.146</td>
<td>0.126</td>
</tr>
<tr>
<td>Over 60</td>
<td>0.416 / 0.031</td>
<td>0.377 / 0.016</td>
<td>0.248 / 0.249</td>
<td>0.125</td>
<td>0.140</td>
</tr>
</tbody>
</table>

(b) Educational background of annotators

<table>
<thead>
<tr>
<th>Education Level</th>
<th>some</th>
<th>hs</th>
<th>bachelors</th>
<th>masters</th>
<th>doctorate</th>
<th>Pessimistic</th>
<th>Optimistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>some</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.085</td>
<td>0.210</td>
</tr>
<tr>
<td>hs</td>
<td>0.116 / 0.738</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.096</td>
<td>0.195</td>
</tr>
<tr>
<td>bachelors</td>
<td>0.109 / 0.790</td>
<td>0.059 / 0.341</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.098</td>
<td>0.193</td>
</tr>
<tr>
<td>masters</td>
<td>0.141 / 0.520</td>
<td>0.070 / 0.378</td>
<td>0.102 / 0.040</td>
<td>-</td>
<td>-</td>
<td>0.076</td>
<td>0.216</td>
</tr>
<tr>
<td>doctorate</td>
<td>0.175 / 0.827</td>
<td>0.217 / 0.407</td>
<td>0.199 / 0.516</td>
<td>0.231 / 0.346</td>
<td>-</td>
<td>0.109</td>
<td>0.190</td>
</tr>
<tr>
<td>professional</td>
<td>0.156 / 0.597</td>
<td>0.104 / 0.134</td>
<td>0.070 / 0.530</td>
<td>0.124 / 0.074</td>
<td>0.184 / 0.647</td>
<td>0.109</td>
<td>0.190</td>
</tr>
</tbody>
</table>

of annotators. While it comprises annotations from 1,708 annotators, the pessimistic subset contains only 1,312. As we can see, this difference is in line with the finding that the annotators of the Wulczyn dataset tended to annotate more liberally.

6 Discussion

The first part of our study addressing RQ1 shows that the proposed approach based on the pessimistic and optimistic scores helps to measure and visualize the difference in the annotation behavior of annotators. In contrast to the inter-rater reliability, our method reveals information about the tendency of the annotators: Did they annotate more liberally or stricter than the group average? These findings can be used to understand outliers better, instruct single annotators in the right direction to align them with the rest of the group and/or adapt the annotation guidelines. Our approach comprises a range of methods, from an aggregated perspective on all annotations to cluster analyses to evaluations of individual annotators. This variety allows handling of datasets with different numbers of annotators.

The proposed approach is unsupervised by itself because it does not require any labeled data. But it can be combined with additional data, as shown by the experiment with the demographic features. We showed that it can help to detect annotator bias caused by different demographic backgrounds. Our results are partially in line with the findings from Al Kuwatly et al. (2020), who examined the same dataset but with a different approach. We confirmed the differences between native and non-native speakers and between the age groups. In our case, we identified a significant difference between male and female annotators, which Al Kuwatly et al. (2020) did not find. In contrast to our experiment, they observed a greater difference between educational backgrounds. The reason for the discrepancy can be the different methods. They trained classifiers on different subsets and compared their performances, as we did for the second part of our study. Furthermore, they had to group the educational backgrounds to have enough data. Consequently, the results can differ. The advantage of our approach over the classifier-based method used by Al Kuwatly et al. (2020) and by Binns et al. (2017) on another dataset is that we do not rely on a classifier as we can use the full dataset.

The underlying assumption for the first part of the study is that there is only one foundational truth whether a text is abusive or not to demonstrate that we all share the same understanding. In the second part of the study, we had the controversial assumption that there are different perspectives on the perception of abusive language. Our goal was to use our proposed method to identify different perspectives and to visualize the differences. By splitting the annotators according to the ratio between the pessimistic and optimistic scores and training different classifiers for these annotators subsets, we showed that there are different perspectives on abusive language. The classifiers of the pessimistic and optimistic annotator subsets per-
form well on their own test set and poorly on the test set of the other subset. That means that the perception of abusive language within each group is coherent, but it differs from the perception of the other subset.

Multiple perspectives on abusive language should be further investigated. Akhtar et al. (2020), for example, showed that balancing different perspectives in the training set can improve the classification performance. We can also imagine building classification models that demonstrate different perspectives; each group would have a customized model based on the groups’ individual values and perceptions.

7 Conclusion
In this paper, we presented a novel approach to measure and visualize annotator bias purely on their annotation behavior. This approach fosters a better understanding of annotation behavior, detecting outliers, and gaining insights on how to adapt annotation guidelines. Furthermore, we showed that there can be different perspectives on abusive language. Using our proposed approach, we can identify these perspectives and examine the differences.

Resources
The code is available under https://github.com/mawic/annotator-bias-abusive-language.

Acknowledgments
This paper is partially based on a joined work in the context of Christian Widmer’s master’s thesis (Widmer, 2021). This research has been partially funded by a scholarship from the Hanns Seidel Foundation financed by the German Federal Ministry of Education and Research. The paper is based upon work partially supported by Google Cloud Platform research credits.

References


Rules Ruling Neural Networks –
Neural vs. Rule-Based Grammar Checking for a Low Resource Language

Linda Wiechetek
UiT Norgga árktalaš universitehta
Norway

Flammie A Pirinen
UiT Norgga árktalaš universitehta
Norway

Mika Hämäläinen
University of Helsinki,
Rootroo Ltd
Finland

Chiara Argese
UiT Norgga árktalaš universitehta
Norway

Abstract

We investigate both rule-based and machine learning methods for the task of compound error correction and evaluate their efficiency for North Sámi, a low resource language. The lack of error-free data needed for a neural approach is a challenge to the development of these tools, which is not shared by bigger languages. In order to compensate for that, we used a rule-based grammar checker to remove erroneous sentences and insert compound errors by splitting correct compounds. We describe how we set up the error detection rules, and how we train a bi-RNN based neural network. The precision of the rule-based model tested on a corpus with real errors (81.0%) is slightly better than the neural model (79.4%). The rule-based model is also more flexible with regard to fixing specific errors requested by the user community. However, the neural model has a better recall (98%). The results suggest that an approach that combines the advantages of both models would be desirable in the future. Our tools and data sets are open-source and freely available on GitHub and Zenodo.

1 Introduction

This paper presents our work on automatically correcting compound errors in real world text of North Sámi and exploring both rule-based and neural network methods. We chose this error type as it is the most frequent grammatical error type (after spelling and punctuation errors) and twice as frequent as the second most frequent grammatical error (agreement error). It also regards both spelling and grammar as the error is a space between two words, but its correction requires grammatical context.

A grammar checker is a writer’s tool and particularly relevant to improve writing skills of a minority language in a bilingual context, as is the case for North Sámi. According to UNESCO (Moseley, 2010), North Sámi, spoken in the North of Norway, Sweden and Finland, has around 30,000 speakers. It is a low resource language in a bilingual setting, and language users frequently face bigger challenges to writing proficiency as there is always a competing language. (Outakoski, 2013) Developing a reliable grammar checker with a high precision that at the same time covers a lot of errors has therefore been our main focus. Good precision (i.e. avoiding false alarms) is a priority because users get easily frustrated if a grammar checker gives false alarms and underlines correct sentences.

In this paper we focus on the correction of compound errors. This type of errors is easy to generate artificially in the absence of large amounts of error marked-up text, and we have a good amount of manually marked-up corpus for evaluation for this error type. Compound errors (i.e. one-word compounds that are erroneously written as two words) can be automatically inserted by using a rule-based morphological analyser on the corpus and splitting the word wherever we get a compound analysis. Unlike other error types (like e.g. real word errors) they are easily inserted, and existing compounds are seldom errors. In addition, they are interesting from a linguistic point of view as they are proper (complex) syntactic errors and not just spelling errors and serve as an example for higher level tools.

Two adjacent words can either be syntactically related or erroneous compounds, depending on the syntax. In North Sámi orthography, as in the majority languages spoken in the region (Norwegian, Swedish and Finnish), nouns that form a new concept are usually written together. For example, the North Sámi word boazodoalloguovlu ‘reindeer herding area’ consists of three words boazu ‘reindeer’, doallu ‘industry’ and guovlu ‘area’, and thus it is written together as a single compound. The task of our methods is to correct spellings such as
boazodoallu guovlu into boazodoalloguovlu in case the words have been written separately in error.

We develop both a rule-based and a neural model for the correction of compound errors. The rule-based model (GramDivvun) is based on finite-state technology and Constraint Grammar. The neural model is bi-directional recurrent (BiRNN). While the rule-based model has earlier produced good precision, it did not handle unknown compounds well, which is why we were interested in a neural approach. However, neural models depend on large amounts of ‘clean’ data and synthetic error generation (or alternatively marked-up data). Typical for low-resource languages and also North Sámi, the corpora are not clean and contain a fair amount of a variety of different spelling and grammatical errors (see Antonsen 2013). Therefore, efficiently preparing data as to making it available for neural model training is an important part of this paper. In our case, we make use of the existing rule-based tools to both, generate synthetic error data and clean the original data for training. For evaluation, on the other hand, we use real world error data.

Our free and open-source rule-based tools can be found on GiellaLT GitHub. The training data and the neural models are freely available on Zenodo. We hereby want to promote a wider academic interest in conducting NLP research for the North Sámi.

2 Background

Sámi open source rule-based language tools have a long and successful tradition (nearly 20 years) (Trosterud, 2004; Moshagen, 2011; Antonsen and Trosterud, 2011; Rueter and Hämiläinen, 2020). North Sámi is a low-resource language in terms of available corpus data (32.24M tokens raw data). Although there is a fair amount of data, it contains many real errors and only a small amount is marked up for errors. Applying neural approaches for high-level language tasks to low resource languages is an interesting research question due the various limitations of minority language corpora, versus the existing research in the topic in well-resourced, majority languages and artificially constrained setups (Nekoto et al., 2020). Rules have been used and are in a wide-spread use in the context of endangered Uralic languages. There is recent work on grammar checking for North Sámi (Wiechetek et al., 2019a) and spell checking for Skolt Sámi (Trosterud and Moshagen, 2021). Other rule-based approaches to grammar checking are extensively described in Wiechetek (2017).

Before the era of neural models, it was common to use statistical machine translation (SMT) as a method for grammar error correction (Behera and Bhattacharyya, 2013; Kunchukuttan et al., 2014; Hoang et al., 2016). Many recent papers on grammar checking use bi-directional LSTM models that are trained to tag errors in an input sentence. Such methods have been proposed for Latvian (Deksnis, 2019), English (Rei and Yannakoudakis, 2016) and Chinese (Huang and Wang, 2016). Similar LSTM based approaches have also been applied for error correction (Yuan and Briscoe, 2016; Ge et al., 2019; Jahan et al., 2021). Other recent approaches (Kantor et al., 2019; Omelianchuk et al., 2020) use methods that take advantage of BERT (Devlin et al., 2019) and other data-hungry models. While such rich sentence embeddings can be used for English and a few other languages with a large amount of data, their use is not viable for North Sámi.

3 Data

For evaluation and training the neural model we use the SIKOR (2018) (the Sámi International KORpus), which is a collection of texts in different Sámi languages compiled by UiT The Arctic University of Norway and the Norwegian Sámi Parliament. It consists of two subcorpora: GT-Bound (texts limited by a copyright which are available only by request) and GT-Free (the publicly available texts). As a preprocessing step, we run a rule-based grammar checker (Wiechetek, 2012) and remove sentences with potential compound errors, as we cannot automatically ensure whether these errors are real or not. This is needed as we want this data to be fully free of any compound errors as it serves as the target side of our neural model.

Thereafter, we take in each sentence in this error free data and analyse it by a rule-based morphological analyser. When the analyser sees a potential compound word, it indicates the word boundary with a compound (+Cmp#) tag. We use this information to automatically split all compounds identified by the rule-based analyser. This results in a

---

1 https://github.com/giellalt/
2 https://zenodo.org/record/5172095
3 https://gtsvn.uit.no/boundcorpus/orig/sme/
4 https://gtsvn.uit.no/freecorpus/orig/sme/
5 https://github.com/giellalt/lang-sme
parallel corpus of the original sentences as the prediction target and their corresponding versions with synthetically introduced compound errors. Many of the compound boundaries are ambiguous, and the algorithm decides the one used in training data based on heuristics: maximum number of compound boundaries where the splitting will not cause any other modifications of the word stems or other content.

As an additional data source, we use the North Sámi Universal Dependencies treebank (Tyers and Sheyanova, 2017). We parse the corpus with Uralic-NLP (Hämäläinen, 2019) and split the compounds the rule-based morphological analyser identifies as consisting of two or more words in order to synthetically introduce errors. We also run the rule-based morphological analyser and morpho-syntactic disambiguator to add part-of-speech (POS) information to produce an additional data set with POS tags. For the Universal Dependencies data, we use the POS tags provided in the data set.

We then make sure that all sentences have at least one generated compound error and that the only type of error the sentences have is the compound error (no other changes introduced by the rule-based models). We shuffle this data randomly and split it on a sentence level into 70% training, 15% validation and 15% testing. The size of the data set can be seen in Table 1, the sentences were tokenized based on punctuation marks.

| Source tokens | Sentences
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>43,658</td>
</tr>
<tr>
<td>Test</td>
<td>9,356</td>
</tr>
<tr>
<td>Validation</td>
<td>9,355</td>
</tr>
<tr>
<td>Real-world errors</td>
<td>3,291</td>
</tr>
</tbody>
</table>

Table 1: Training, testing and validation sizes for the neural model (corpus with synthetic errors)

For the rule-based model GramDivvun we do not generate synthetic errors. We have hand-selected a large corpus for rule development and as regression tests, consisting of representative sentences from GT-Free. The current selection for syntactic compound errors includes 3,291 sentences with real world compound errors (and possibly other errors in addition).

4 Methods

We use a neural models and a rule-based model for compound error correction.

4.1 Neural Model

We model the problem at a character instead of word level in NMT (neural machine translation). The reason for using a character-level model instead of a word-level model is that, this way, the model can work better with out-of-vocabulary words. This is important due to the low-resourced nature of North Sámi, although there are other deep learning methods for endangered languages that do not utilize character level models (Alnajjar, 2021). In practice, we split words into characters separated by white spaces and mark actual spaces between words with an underscore (_). We train the model to predict from text with compound errors into text without compound errors. As previous research (Partanen et al., 2019; Alnajjar et al., 2020) has found that using chunks of words instead of full sentences at a time improves the results in character level models, we will be training different models with different chunk sizes. This means that we will train a model to predict two words at a time, three words at a time, all the way to five words at a time.

We train the models with and without POS tags. For the models with POS tags, we surround each word with a token indicating the beginning and the end of the POS tag. The POS tags are included only on the source side, not on the target side. They are separated from the word with a white space.

An example of the data can be seen in Table 2. Even though every sentence in the training data has a compound error, this does not mean that every input chunk the model sees would have a compound error. This way, the model will also learn to leave the input unchanged if no compound errors are detected.

We train all models using a bi-directional long short-term memory (LSTM) based model (Hochreiter and Schmidhuber, 1997) by using OpenNMT-py (Klein et al., 2017) with the default settings except for the encoder where we use a BiRNN (Schuster and Paliwal, 1997) instead of the default RNN (recurrent neural network), since BiRNN based models have been shown to provide better results in character-level models (Hämäläinen et al., 2019). We use the default of two layers for both the encoder and the decoder and the default attention model, which is the general global attention presented by Luong et al. (2015). The models are trained for the default of 100,000 steps. All models are trained with the same random seed (3,435) to ensure reproducibility.
During the training of the neural models, we evaluate the models using simple sentence level scores. There we look only at full-sentence matches and evaluate their accuracy, precision and recall, as opposed to the evaluations in Section 5, where we study them more carefully at the word-level. The results of the neural models for the generated corpus (where errors were introduced by splitting compounds) can be seen in Table 3. The results indicate that both of the models receiving a chunk of two words at a time reached to the highest accuracy, and the model without the POS tags also reached to the highest precision.

<table>
<thead>
<tr>
<th>Chunk</th>
<th>POS</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>no</td>
<td>0.925</td>
<td>0.949</td>
<td>0.974</td>
</tr>
<tr>
<td>3</td>
<td>no</td>
<td>0.847</td>
<td>0.883</td>
<td>0.935</td>
</tr>
<tr>
<td>4</td>
<td>no</td>
<td>0.852</td>
<td>0.892</td>
<td>0.930</td>
</tr>
<tr>
<td>5</td>
<td>no</td>
<td>0.869</td>
<td>0.909</td>
<td>0.952</td>
</tr>
<tr>
<td>2</td>
<td>yes</td>
<td>0.925</td>
<td>0.948</td>
<td>0.976</td>
</tr>
<tr>
<td>3</td>
<td>yes</td>
<td>0.906</td>
<td>0.934</td>
<td>0.968</td>
</tr>
<tr>
<td>4</td>
<td>yes</td>
<td>0.856</td>
<td>0.896</td>
<td>0.951</td>
</tr>
<tr>
<td>5</td>
<td>yes</td>
<td>0.857</td>
<td>0.895</td>
<td>0.935</td>
</tr>
</tbody>
</table>

Table 3: Sentence level scores for different neural models tested on a corpus with artificially introduced errors.

The POS tags were not important for the models, as the results with and without them are fairly similar. The largest gain was when the compound error correction was done for three words at a time. As this performance gain only occurred for that specific model, it suggests that it is more of an artefact of the training data and how it is fed into the model than any actual improvement.

### 4.2 Rule-based Model

The rule-based grammar checker GramDivvun is a full-fledged grammar checker fixing spelling errors, (morpho-)syntactic errors (including real word spelling errors\(^6\), inflection errors, and compounding errors) and punctuation and spacing errors.

It takes input from the finite-state transducer (FST) to a number of other modules, the core of which are several Constraint Grammar modules for tokenization disambiguation, morphosyntactic disambiguation and a module for error detection and correction. The full modular structure (Figure 1) is described in Wiechetek (2019b). This work regards predominantly the modification of the disambiguation and error detection modules mwe-dis.cg3, grc-disambiguator.cg3, and grammarchecker-release.cg3. We are using finite-state morphology (Beesley and Karttunen, 2003) to model word formation processes. The technology behind our FSTs is described in Pirinen (2014). Constraint Grammar is a rule-based formalism for writing disambiguation and syntactic annotation grammars (Karlsson, 1990; Karlsson et al., 1995). In our work, we use the free open source implementation VISLCG-3 (Bick and Didriksen, 2015). All components are compiled and built using the GiellaLT infrastructure (Moshagen et al., 2013). The code and data for the model is available for download\(^7\) with specific version tagged for reproducibility.

The syntactic context is specified in hand-written Constraint Grammar rules. The REMOVE-rules below removes the compound error reading (identified by the tag Err/SpaceCmp) if the head is a 3rd person singular verb (cf. 1.2) and the first element of the potential compound is a noun in nominative case (cf. 1.3). The context condition further specifies that there should be a finite verb (VFIN) somewhere in the sentence (cf. 1.4) for the rule to apply:

```
REMOVE (Err/SpaceCmp)
(0/0 (V Sg3))
(0/1 (N Sg Nom))
(+0 VFIN);
```

All possible compounds written apart are considered to be errors by default, unless the lexicon specifies a two or several word compound or a syntactic rule removes the error reading.

\(^6\)Real word errors are spelling errors where the outcome is an actual word that is not fit for the context.

\(^7\)https://github.com/giellalt/lang-sme/releases/tag/naacl-2021-ws
The process of rule writing includes several consecutive steps, and like neural network models they require data. The process is as follows:

1. Modelling an error detection rule based on at least one actual sentence containing the error
2. Adding constraints based on the linguist’s knowledge of possible contexts (remembered data)
3. A corpus search for sentences containing similar forms/errors, testing of the rule and reporting rule mistakes
4. Modification of constraints in the rule based on this data and testing against regression tests so that unfit constraints depending on results for precision and recall (focus on precision)

The basis of rule development is continuous integration. Typical shortcomings and bad errors can be fixed right away with added conditions. Neural models are not usually trained in this way. The frequent experience of false alarms can decrease the users’ trust in the grammar checker. Typically, full-fledged user oriented grammar checkers, e.g. DanProof focus on keeping false alarms low and precision high (Bick, 2015) because users’ experiences have shown that certain experiences will frustrate users and stop them from using the application.

For rule development, regression tests are used. These consist in error-specific YAML tests and are manually marked up. The regression test for compound errors contains 3,291 sentences (1,368 compound errors, used for development and regression) give the results as shown in Table 4.

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>F₁ score</th>
</tr>
</thead>
<tbody>
<tr>
<td>94.95</td>
<td>86.22</td>
<td>90.80</td>
</tr>
</tbody>
</table>

Table 4: The rule-based model tested on the developer’s corpus (regression tests)

5 Results

We evaluate the models both quantitatively and qualitatively. We evaluate on accuracy, precision and recall, and do a linguistic evaluation. The measurements are defined in this article as follows: Accuracy \( A = \frac{C}{S} \), where \( C \) is a correct sentence (1:1 string match) and \( S \) is corpus size in sentences, precision \( P = \frac{tp}{tp+fp} \) and recall \( R = \frac{tp}{tp+fn} \), where \( tp \) is true positive, \( fp \) is false positive and \( fn \) is false negative. The \( F₁ \) score is the harmonic mean of precision and recall \( F₁ = \frac{2 \times P \times R}{P+R} \). The accuracy is thus sentence level correctness rate—as used in the method section to probe model qualities—whereas precision measures how often corrections were right and recall measures how many errors we found. The word-level errors are counted once per error in the marked-up corpus. Thus, if a three-part compound contains two compounding errors it is counted towards the total as one error, but if a sentence has three separate compounds with wrong splits each, we count three errors.
The error marked-up corpus we used includes 140 syntactic compound errors (there are other compound errors that can be discovered by the spellchecker as they are word internal) and is from GT-Bound. We chose GT-Bound to make sure that the sentences had not been used to develop rules. It is part of our error-marked up corpus, which makes it possible to run an automatic analysis. This error corpus does only contain real world (as opposed to synthetic) errors.

Table 5: Sentence level scores for the neural models tested on a real world error corpus

Table 5 shows the results for the neural models on this corpus. The drop in results is expected as the models were trained on synthetic data, whereas this data consists of real world errors. However, the results stay relatively good, given that synthetic data was the only way to produce enough training data for North Sámi.

We ran the neural and rule-based model on two different corpora of compound error materials, i.e. synthetic and real world. Table 6 shows the evaluation on a real world error corpus.

Table 6: Results for both models based on a manually marked-up evaluation corpus

The neural network performs well in terms of numbers, but has the following shortcomings that are problematic for the end users. It introduces new (types of) errors unrelated to compounding, like changing \(km^2\) randomly either to \(kmy\) or \(km\) kind of unforgivable (because not understandable) for the end user. They introduce compounds like Statoileamiálbmogiid ‘Statoil (national oil company and gasstation) indigenous people’ as in ex. (1). The rule-based grammar checker presupposes that the compound is listed in the lexicon, which is why these corrections can easily be avoided.

1. **Statoil eamiálbmogiid**

   Statoil indigenous.people.ACC.PL
eatnamiid billisteani birra
land.ACC.PL. destruction GEN
about
‘about the destruction of the indigenous peoples’ territories by Statoil’

It also produces untypically long non-sense words like NorggaSámiiRiidRiidRiidRiidRiid

2. **boares eallinoainnuid** ja modearna

   old.life view.ACC.PL and modern
   servodaga váikkuhusaid gaskii.
society. GEN impact.ACC.PL between
   ‘between old philosophies and the impact of modern society’

3. **Dasalassin 137000 njealjehaskilomehtera eatnamat** in.addition 137000 square.kilometre. GEN

   eatnamat bidgejuvvojit seismalas
   land.PL. split.PASS.PL.3 seismic
   linnjáid line.ACC.PL
   ‘In addition, 137,000 square kilometres of land are split by seismic lines’

The rule-based model, on the other hand, typically suggests compounding, where both compounding and two word combinations would be adequate, for example in the case of the first part of the compound having homonymous genitive and a nominative analyses. The suggested compound is not an error. However, the written form is grammatically correct as well. These suggestions still count as false positives. Other typical errors are cases where there are two accepted ways of spelling a compound/MWE as in ex. (4), where both Riddu Riddu and Riddu-Riddu are correct spellings, and the latter one is suggested as a correction of the former one.

4. **ovdanbuktojuvvojit omd. jahkásaš** present.PASS.PRS.PL.3 e.g. annual

   Riddu Riddu festiválas.
   Riddu Riddu festival.PL.
   ‘they are presented at the annual Riddu Riddu festival.

The rule-based model also struggles predominantly
with false negatives, like *njunuš olbmot* ‘leading people’ that are due to missing entries in the lexicon like in ex. (5).

(5) *Sii leat giellda njunuš olbmot.*

They are the leading people of the municipality.

6 Discussion

In the future, we would like to look into hybrid grammar checking of other error types and other (Sámi) languages.

The neural approach gives us relatively high recall in the real world situation with lower precision, whereas the rule-based model is designed to give us high precision even at the cost of lower recall (user experience), which is why hybrid approaches that combine the best of two worlds are interesting.

Noisy data is to be expected in any endangered language context, as the language norms are to a lesser degree internalized. We will therefore need a way of preparing the data to train neural networks, which can either consist in creating synthetic data or automatically fixing errors and creating a parallel corpus.

When creating synthetic data for neural networks, the amount of data is hardly the main issue. Many generative systems are capable of overgenerating data. The main question that arises is the quality and representatives (Hämäläinen and Alnajjar 2019) of the generated data. If the rules used to generate the data are not in line with the real world phenomenon the neural model is meant to solve, we cannot expect very high quality results in real world data.

Generated sentences can easily be less complex ‘text book examples’ that are not representative of real world examples. In the case of agreement errors between subjects and verbs, for example, there are long distance relationships and complex coordinated subjects including personal pronouns that can change the structure of a seemingly straightforward relation. Therefore, we advocate the use of high quality rule-based tools to prepare the data, i.e. fix the errors and create a parallel corpus.

While synthetic error data generation for compound errors is somewhat more straightforward as it only affects adjacent words, the generation of synthetic error corpora for other error types is not as straightforward, in part also because generating synthetic errors of other kind can potentially create valid and grammatically correct sentences with different meanings. We therefore predict that (hybrid) neural network approaches for other error types that either involve specific morphological forms (of which there are many in North Sámi) or changes in word order will be more difficult to resolve.

7 Conclusion

In this paper, we have developed both a neural network and a rule-based grammar checker module for compound errors in North Sámi. We have shown that a neural compound-corrector for a low-resource language can be built based on synthetic error data by introducing the compound errors using a high level rule-based grammar models. It is based on the rule-based tools to both generate errors and clean the data using both part-of-speech analysis, disambiguation and even the error detector.

The rule-based module is embedded in the full-fledged *GramDivvun* grammar checker and achieves a good precision of 81% and a lower recall of 61%. A higher precision, even at the cost of a lower recall, is in line with our objective of keeping false alarms low, so users will be comfortable using our language tools. The neural network achieves a slightly lower precision of 79% and a much higher recall of 98%.

However, the rule-based model has more user-friendly suggestions and some false positives are simply other correct alternatives to the ones in the text, while the neural network’s false positives sometimes introduce new and unrelated errors. On-the-fly fixes that avoid false positives are an advantage of rule-based models. Rule-based models, on the other hand, are not so good at recognizing unknown combinations. Hybrid models that combine the benefits of both approaches are therefore desirable for efficient compound error correction in the future.

Acknowledgments

Thanks to Børre Gaup for his work on the evaluation script. Some computations were performed on resources provided by UNINETT Sigma2 – the National Infrastructure for High Performance Computing and Data Storage in Norway.
References


Duc Tam Hoang, Shamil Chollampatt, and Hwee Tou Ng. 2016. Exploiting n-best hypotheses to improve an smt approach to grammatical error correction. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, pages 2803–2809.


Transformer with Syntactic Position Encoding for Machine Translation
Yikuan Xie\textsuperscript{1}, Wenyong Wang\textsuperscript{1}, Mingqian Du\textsuperscript{2}, and Qing He\textsuperscript{3}

\textsuperscript{1}University of Electronic Science and Technology of China
\textsuperscript{2}The Second Research Institute of CAAC
\textsuperscript{3}The Third People’s Hospital of Chengdu City
xieyikuan@std.uestc.edu.cn, wangwy@uestc.edu.cn
dumingqian@caacetc.com, 123hq@163.com

Abstract

It has been widely recognized that syntax information can help end-to-end neural machine translation (NMT) systems to achieve better translation. In order to integrate dependency information into Transformer based NMT, existing approaches either exploit words’ local head-dependent relations, ignoring their non-local neighbors carrying important context; or approximate two words’ syntactic relation by their relative distance on the dependency tree, sacrificing exactness. To address these issues, we propose global positional encoding for dependency tree, a new scheme that facilitates syntactic relation modeling between any two words with keeping exactness and without immediate neighbor constraint. Experiment results on NC11 German$\rightarrow$English, English$\rightarrow$German and WMT English$\rightarrow$German datasets show that our approach is more effective than the above two strategies. In addition, our experiments quantitatively show that compared with higher layers, lower layers of the model are more proper places to incorporate syntax information in terms of each layer’s preference to the syntactic pattern and the final performance.

1 Introduction

Over the past few years, end-to-end neural machine translation (NMT) has shown remarkable progress, achieving promising results on various machine translation tasks. Although NMT models can perform well even only trained on parallel corpus, they have been found to still benefit from external linguistic knowledge (e.g., lexical analysis and parsing) (Sennrich and Haddow, 2016), and the same finding was also observed in other tasks (Strubell et al., 2018; Zhang et al., 2019). Therefore, several studies have started to to investigate how to incorporate dependency information into NMT based on Transformer, which is currently the most advanced end-to-end machine translation backbone.

There exist two broad categories of approaches to integrating a sentence’s dependency structure into the Transformer. The first category of approaches (Bugliarello and Okazaki, 2020; Strubell et al., 2018; Bastings et al., 2017) takes the dependency tree as a general graph structure data. A graph neural network or a sublayer with a similar mechanism was introduced at the bottom of the Transformer’s encoder to represent and encode the dependency relations among the words in the sentence. In this paradigm, however, a word focuses only on its immediate neighbors and ignores nodes that are multi-hop away from it, which may carry important context that helps disambiguate and enrich the current word’s representation in terms of syntactic structure.

In contrast, methods from the second category seek to model two words’ syntactic relation relying on their distance from the dependency tree. The distance can be two words’ relative depth on the tree or the length of the path between the two nodes (Wang et al., 2019a; Omote et al., 2019). This strategy can be seen as extended relative position representations (Shaw et al., 2018) where the relative distance is defined based on the tree. The advantage of this strategy is that the syntactic relations between a word and all other words can be modeled, not just those of its immediate neighbors. Nevertheless, this approach is also unsatisfactory because the distance on the dependency tree is a simple proxy for the syntactic relationship between two words and does not represent how a word reaches another word on the dependency tree. Therefore, this type of approach has limitations in expressiveness.

For these considerations, we propose to model the position of the word on the dependency tree in addition to its normal sequential position in the input text. We use the path from the root node to a
word on the dependency tree to represent the global syntactic position of the word. This representation facilitates the modeling of syntactic relations between two words at arbitrary distances while enabling the model to learn more powerful correlation functions by exploiting dependency labels rather than simply relying on the distance of word pairs.

Our main contributions are as follows:

- We propose a new scheme, global positional encoding for syntax (GPS), to integrate the dependency tree into the Transformer’s encoder, enabling efficient word pairs’ syntax correlation modeling (Section 3).
- Experimental results demonstrate that our method outperforms both the Transformer baseline and other two competitive models in terms of BLEU score (Section 4.2).
- We found that when external syntax annotation is combined at lower layers of the encoder, it gives better results and has larger weights; and the weights, which indicate a layer’s reliance on syntactic information, declines rapidly near the output layer of the encoder (Section 4.3).

## 2 Related Work

Previous studies showed that end-to-end NMT models could be further improved by combining source-side or target-side syntax structure into RNN encoder-decoder framework (Aharoni and Goldberg, 2017; Wu et al., 2017; Eriguchi et al., 2016). Since the Transformer has become the de facto standard architecture for NMT, researchers have also begun to explore strategies to incorporate syntax information into Transformer-based NMT.

For phrase structures, Currey and Heafield (2019) used source-side linearized constituency parses to improve Transformer-based NMT by multi-task learning. Ma et al. (2019) used neural syntactic distance (Shen et al., 2018) for constituent parsing as input and output sequence.

For dependency grammar, researchers mainly focused on how to integrate the dependency structure into the Transformer. As mentioned in the introduction, there are two different viewpoints to look upon this problem. The first line of works extended the relative positional encoding (Shaw et al., 2018) defined on sequence to tree structures. Omote et al. (2019) defined the relative distance between two words on the dependency tree in terms of their relative depth. Wang et al. (2019b) used the length of the shortest path between two nodes on the dependency tree as their relative distance, and the depth of each node as their absolute position, in addition to each word’s normal sequential position. Another perspective is to take the dependency tree as a general graph structure, focusing on local connection relations and apply methods from graph neural networks (or use similar mechanisms). Bastings et al. (2017) use a graph-convolutional networks to encode the source syntax structure, and more recently, Bugliarello and Okazaki (2020) directly encode the dependency structure inside the Transformer’s self-attention module, achieving state-of-the-art results.

By contrast, we neither treat the dependency tree as a simple sequential structure extension nor as general graph data. We seek to represent each word’s role on the dependency tree, where the representation is tailored for the dependency grammar’s tree structure, making it sensible to the syntactic connection with other words.

Our work is also inspired by researches on making Transformer sensible to structured input. Shiv and Quirk (2019) proposed a tree-structure positional encoding to make the Transformer sensible to source code’s tree structure in the code processing task. The main idea is to define the path from the root node to a leaf node as the leaf node’s position on the tree. We extend this idea from unlabeled directed trees to labeled trees and break the limitation of the tree depth and the number of child nodes. Moreover, we choose to use a sequence encoder instead of concatenating sub-vectors to compute the final path embedding, which utilizes the whole vector space and enables more powerful modeling of the interaction between two path embeddings.

## 3 Model

In this section, we first recap the basic Transformer architecture and then describe the proposed global syntactic position encoding mechanism.

### 3.1 Transformer Architecture

Transformer (Vaswani et al., 2017) is an encoder-decoder architecture composed of stacked encoder and decoder layers. Each encoder layer has three main components: residual connections, feed-forward layer, and self-attention. The decoder layer has extra decoder-encoder attention to access the output of the encoder. Among these components,
the self-attention mechanism is the key feature that sets Transformer apart from CNN-based and RNN-based sequence to sequence (seq2seq) models. Self-attention is used for improving a word’s representation by focusing on critical context words that help to enrich and disambiguate it.

Given an input sequence \( x = \{x_0, ..., x_n\} \), self-attention outputs a new sequence \( z = \{z_0, ..., z_n\} \). Each word’s representation in the new sequence are contextualize by a weighted sum over all words in the old sequence.

\[
z_i = \sum_j^n \alpha_{ij} (x_j W^V)
\]  

Each weight, \( \alpha_{ij} \), is normalized by a softmax function:

\[
\alpha_{ij} = \frac{\exp(e_{ij})}{\sum^n_{k=1} \exp(e_{ik})}
\]  

\( e_{ij} \) is computed by the dot product of \( word_i \)’s query vector and \( word_j \)’s key vector

\[
e_{ij} = \frac{(x_i W^Q) (x_j W^K)^T}{\sqrt{d_z}}
\]  

where \( \sqrt{d_z} \), a scale factor, reduces the peak of the attention distribution to alleviate the vanishing gradients problem.

### 3.1.1 Sequential Positional Encoding

Transformer’s self-attention can access information directly from other words no matter how far away they are from the current word, making it easier to model long-range dependencies. However, unlike RNN’s intrinsic order-variant forward process, the self-attention module is computationally insensible to the word order. Thus we need to put the word order information into the representation of the input sequence. In practice, we often simply add the positional embedding and input word embedding together as inputs to the model. Transformer’s default positional embeddings has the following formulation:

\[
PE_{(pos,2i)} = \sin \left( \frac{pos}{10000^{2i/d_{model}}} \right) \\
PE_{(pos,2i+1)} = \cos \left( \frac{pos}{10000^{2i/d_{model}}} \right)
\]  

where \( pos \) is the index of a word in the sequence and \( i \) is the dimension of the positional encoding vector.

### 3.2 Positional Encoding for Dependency Structure

A sentence exists as a sequence of words, but it also has structural organization. Dependency parsing describes a sentence as a labeled directed tree, whose vertices correspond to words, and labeled edge denotes head-dependent relation between words. Besides each word’s sequential position, we would like to represent their positions on the dependency tree to exploit prior linguistic knowledge behind the sentence.

We follow similar steps of encoding position in sequential structure to represent a word’s position on the dependency tree. We need first to index a word and then transform the index into a vector, also called positional embedding, and the computation between two positional embeddings should reflect their relationship on the target structure. For sequential structure, the relation is the distance, and for the dependency tree, the relation is the grammatical relation denoted by how a node reaches another node. Therefore, we can represent
a words’ tree position through the path from the root node to it. Figure 1 showing an example, John can be represented by a sequence of grammatical labels \([\text{root} \rightarrow \text{conj} \rightarrow \text{cc}]\) and school can be represented by \([\text{root} \rightarrow \text{obl}]\).

In this setting, the syntactic relation between two words can be inferred from their position representation on the syntactic tree.

To acquire the final syntactic positional embedding, we only need to vectorize the path. First, we replace all the labels with index from the dependency label vocabulary, then we encode the sequence into a vector, as if we are dealing with a sequence of words. Any neural network that can transform a sequence into a vector fits this job well, and we take LSTM as our syntactic position encoder:

\[ s_i = LSTM(p_i) \]

where \(s_i\) is the syntactic positional embedding for \(x_i\), and \(p_i = \{\text{label}_0, \ldots, \text{label}_n\}\) is the path from root to \(x_i\) on dependency tree.

### 3.3 Combining the Two Positional Embeddings

Once we can represent a word’s position on the dependency tree, the next question is how to enrich a word’s hidden representation by its position on the syntactic tree, in addition to its position in the sequence. Following Transformer’s initial setup to incorporate sequential positional embedding, we may add the syntactic positional embedding into the input word embedding. However, this approach suffers from two drawbacks. First, when summing up heterogeneous embeddings, we are implicitly model correlation between different types of embedding in the self-attention module, which is noticed by Ke et al. (2021). To realize the problem here, we can split the hidden representation of a word \(x_i\) into the word embedding \(w_i\), positional embedding \(p_i\), and syntactic positional embedding \(s_i\), and see how the self-attention computes the attention weight:

\[
\hat{e}_{ij} = \frac{(w_i + s_i + p_j)W^Q}{\sqrt{d_z}} - \frac{(w_j + s_j + p_i)W^K}{\sqrt{d_z}} \tag{4}
\]

After expanding the above equation, we can see that the attention weight of word \(i\) to word \(j\), denoted by \(e_{ij}\), consists of the correlation between heterogeneous embeddings, e.g., the correlation between syntactic positional embedding and sequential positional embedding, which is not reasonable.

More importantly, when a source sequence contains multiple sentences, mixing the syntactic positional embeddings with word embedding allows two words from different sentences to interact with each other syntactically in the self-attention module. However, two words’ syntax annotations are supposed to affect each other only within a sentence.

Therefore, we decouple the syntactic positional embedding and word embedding:

\[
\hat{e}_{ij} = \frac{(x_iW^Q) (x_jW^K)^T}{\sqrt{d_z}} + \frac{(s_iW^Q_s) (s_jW^K_s)^T}{\sqrt{d_z}} \tag{5}
\]

\[
x_i = w_i + p_i \tag{6}
\]

where \(x_i\) is the word representation as in Transformer’s typical setting, and \(s_i\) denotes the syntactic positional embedding of word \(i\). In this manner, we can eliminate the unnecessary correlation between heterogeneous embeddings and avoid undesired syntactic correlation for two words from different sentences by simply mask the second term in equation (5).

### 4 Experiment

#### 4.1 Experimental Setup

**Model and Baselines.** We compare our approach with a strong baseline, Transformer, and other two types of dependency information augmented strategies: (1) Parent scaled self-attention (PASCAL) proposed by Bugliarello and Okazaki (2020), in which a word’s attention distribution is scaled to have larger weight on its head word. (2) Absolute and relative structural position proposed by Wang et al. (2019b), which augment encoder by each word’s depth and word-pair’s relative distance on the dependency tree. We implement our models and reimplement Transformer with structural position (Wang et al., 2019b) based on the Fairseq (Ott et al., 2019) toolkit. We deploy the syntactic positional encoding at the bottom layer for our model and report the performance in the main results section. See section 4.3 for further analysis of this design choice.

**Data.** We use datasets with different sizes and source languages to evaluate the efficiency of our
We can see from Table 2 that in all three syntax-augmented NMT systems, syntax information improves the performance of the baseline model at least 0.5 BLEU points.

On the other hand, although the distance-based approach (Structure Position) can theoretically model the syntactic relationship between any pair of words, it performs slightly inferior (0.1–0.3 BLEU points) than the PASCAL model, which only represents local head-dependent relations; this may be caused by information loss when using distance to approximate the syntactic relation between two words. In contrast, our strategy represents each word’s global position on the syntax tree, facilitating efficient syntax relation modeling between non-local word pairs on the dependency tree. This further leads to consistent improvements over other models.

4.2 Effects of Syntax at Different Layers

Previous researches empirically found that combining the syntax information at the bottom layer of the encoder gives the best result (Strubell et al., 2018; Bugliarello and Okazaki, 2020). In this subsection, we measure how our model’s performance changes over the layer that we choose to combine the syntactic position, and quantify the syntax’s impact on each layer’s self-attention module.

First, we change the layer used to combine the syntactic position and train the model on the NC11 De-En dataset. We present the performance over the selected layer in Table 3. Consistent with previous studies, we observed that combining syntax structure at lower layers gives better results and achieves the best at the bottom layer. However, though the syntactic information brings performance gains no matter at which layer we combine it, these improvements are not complementary. When we combine the syntactic position at all six layers of the encoder, we observe comparable results than only combining it at the first layer of the encoder while accompanied by severe overfitting and much more training time.
<table>
<thead>
<tr>
<th>Model</th>
<th>NC11 En-De</th>
<th>NC11 De-EN</th>
<th>WMT16 En-De</th>
<th>WMT17 En-De</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer</td>
<td>25.0</td>
<td>26.6</td>
<td>33.0</td>
<td>25.5</td>
</tr>
<tr>
<td>PASCAL</td>
<td>25.9</td>
<td>27.4</td>
<td>33.9</td>
<td>26.1</td>
</tr>
<tr>
<td>Structure Position</td>
<td>25.7</td>
<td>27.1</td>
<td>33.8</td>
<td>26.0</td>
</tr>
<tr>
<td>Ours</td>
<td>26.0(\dagger)</td>
<td>27.6(\dagger)</td>
<td>34.1(\dagger)</td>
<td>26.3(\dagger)</td>
</tr>
</tbody>
</table>

Table 2: Performances on the test set of three datasets in terms of BLEU score, with \(\dagger\) denotes statistical significance over the Transformer \((p < 0.05)\) through boot strap resampling (Koehn, 2004).

In our setting, the syntactic information exerts influence directly on the self-attention distribution. Nevertheless, it remains unclear to what extent the model relies on the supplied syntax information, whose answer would provide us more interpretability and understanding of what is happening inside the model. To figure out this problem, we define the Average Weight of Syntax Logits (AWSL) as a metric to indicates the weight of syntax in the self-attention module. For each sample and attention head, we compute the weight of syntax logits as:

\[
w = \frac{1}{N^2} \sum_i \sum_j |e_{ij}| + |e_{ij}^s|
\]

where \(N\) is the number of words in the sentence, \(e_{ij}\) and \(e_{ij}^s\) denotes the normal attention weight and the syntax correlation weight of word \(i\) to word \(j\), respectively. We then average the weight \(w\) for each sample and attention head to compute \(AWSL_L\) for layer \(L\), as plotted in Figure 3.

Through the lens of AWSL, we can observe that the model relies on the abstract syntax patterns more at the lower layers. Such reliance becomes gradually smaller as the layer becomes deeper but drops sharply near the output layer.

One possible explanation for the two trends about model performance and feature importance is that combining syntax information at lower layers not only leads to better utilization across all subsequent layers but also reaches a closer match in terms of the abstraction level between the layer and the supplied information. We note that this finding is in line with previous studies on probing the hidden representations of the deep language model (Raganato and Tiedemann, 2018; Tenney et al., 2019; Vig and Belinkov, 2019), which found that the model tends to encode more syntactic features in lower layers while more complex semantic features in higher layers.
Table 3: Model performance over the layer chose to combine syntactic position.

<table>
<thead>
<tr>
<th>Layer</th>
<th>None</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>1-6</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLEU</td>
<td>23.8</td>
<td>24.7</td>
<td>24.5</td>
<td>24.5</td>
<td>24.0</td>
<td>24.1</td>
<td>24.0</td>
<td>24.7</td>
</tr>
</tbody>
</table>

Table 4: We can see through the two cases that the baseline model generates syntactically incorrect translation due to the misunderstanding of the source sentence’s syntactic structure, and the supplied syntax structure helps it achieve more faithful translation.

4.4 Case Study

Table 4 presents several representative cases that indicate how syntactic knowledge helps the model achieve better translation, while linguistically-uninformed baseline makes mistakes. In the first case, the meeting is the object to be rejected. However, the baseline takes meeting as nsubj (nominal subject) of the verb rejected, while our model correctly identified the meeting as the verb’s object. A more complex example is the second case, where the adjective anfällig (means fragile) governs the nominal Neuronen (means neurons). The baseline wrongly and indirectly connects therapy and vulnerable, establishing misleading connection “therapy is made vulnerable”, while our model correctly set up the second-order relation between neurons and vulnerable: “make neurons vulnerable”.

5 Conclusion and Future Work

In this paper, we propose a global position encoding scheme for the dependency tree. We leverage the dependency labels and represent each word’s syntactic role on the dependency tree using the path, which enables efficient syntactic correlation modeling between any pair of words in a single layer. In the experiment, we show that our model outperforms other syntax augmentation strategies. In addition, we quantitatively analyze the model’s reliance on syntax information and show that the model pay more weight and achieves better performance when we combine syntax information at lower layers.

For future works, the following problems are worth noting:

1. Since it is commonly known that off-the-shelf parser’s performance may drop dramatically when facing out-of-domain data, it is important to assess the parser’s accuracy on different machine translation benchmarks and evaluate those syntax augmented NMT systems’ tolerance of parsing errors.

2. Previous works have shown that human-
designed patterns are not the only option to establish dependency relations among words; trees induced from the Pre-trained Language Models also exhibit promising results in downstream applications (Wu et al., 2020; Dai et al., 2021). The induced tree is an attractive solution, especially when considering the expensive annotation and domain adaption problem for supervised dependency parser. Therefore, it is desirable to compare the performance of model using trees produced by the supervised parser and trees induced from the Pre-trained Language Models.

6 Acknowledgement

The work was supported by key project of NSFC-Civil Aviation Joint Fund(U2033212). The authors thank the three anonymous reviewers for their valuable comments.

References


Towards Sentiment Analysis of Tobacco Products’ Usage in Social Media

Venkata Himakar Yanamandra∗, Kartikey Pant∗ and Radhika Mamidi
International Institute of Information Technology, Hyderabad, India
{himakar.y, kartikey.pant}@research.iiit.ac.in, radhika.mamidi@iiit.ac.in.

Abstract
Contemporary tobacco-related studies are mostly concerned with a single social media platform while missing out on a broader audience. Moreover, they are heavily reliant on labeled datasets, which are expensive to make. In this work, we explore sentiment and product identification on tobacco-related text from two social media platforms. We release SentiSmoke-Twitter and SentiSmoke-Reddit datasets, along with a comprehensive annotation schema for identifying tobacco products’ sentiment. We then perform benchmarking text classification experiments using state-of-the-art models, including BERT, RoBERTa, and DistilBERT. Our experiments show F1 scores as high as 0.72 for sentiment identification in the Twitter dataset, 0.46 for sentiment identification, and 0.57 for product identification using semi-supervised learning for Reddit.

1 Introduction
Smoking tobacco causes more than 8 million deaths each year1. While cigarette smoking has fallen globally, e-cigarettes remain the commonly used tobacco product among the US youth2.

So far, 2,807 cases and 68 deaths have been recorded due to e-cigarette use-associated lung injury in the US2 alone. These developments prompted the US government to enforce a countrywide ban on addictive flavors and sales to minors. As smokers are more likely to develop severe COVID-19 symptoms, WHO has introduced the world’s first digital health worker Florence based on AI to fill the gap between overburdened medical system users who are trying to quit3. Health-related discourses on social media have risen substantially over the years (Tamersoy et al., 2015) since social media provides an opportunity to informally express opinions freely with like-minded individuals across geographical and societal barriers.

Twitter boasts of 330 million active monthly users who send out half a billion tweets daily4 while Reddit has 430 million monthly users and has 100k+ communities 5. Reddit provides expendable and pseudo-anonymous accounts that are well suited for controversial discussions, including the perception of electronic cigarettes and marijuana, which might be inappropriate to discuss on non-anonymous forums (Park and Conway, 2018).

In Salathé and Khandelwal (2011), we see that information flows more often between users who share the same sentiments. Studying trends of first-person accounts of experience and sentiment towards different tobacco products in these forums becomes imperative in the ongoing bio-surveillance and regulatory efforts (Kim et al., 2015; Pant et al., 2019). Real-time monitoring of public sentiment and informativeness gives us an opportunity for bottom-up discovery of emergent patterns, especially in vulnerable and ethnically diverse populations that may not be readily detectable by traditional methods (Myslín et al., 2013; Lienemann et al., 2017). Although Yanamandra et al. (2020) identified tobacco products on Twitter, they leave a gap in identifying the products’ sentiment.

This work explores tobacco products’ identification and their sentiments in texts extracted from two commonly-used social media platforms. We release two datasets for multiclass classification annotated with tobacco products and their sentiment: SentiSmoke-Twitter and SentiSmoke-Reddit. 6 We

2https://www.cdc.gov/tobacco/
5https://www.redditinc.com/
6https://github.com/himakaryv/SentiSmoke-Datasets
**Table 1: Examples from the Dataset for each class and source.**

<table>
<thead>
<tr>
<th>Class</th>
<th>Source</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>Twitter</td>
<td>&quot;Cigarette a day keeps the depression away&quot;</td>
</tr>
<tr>
<td></td>
<td>Reddit</td>
<td>&quot;I just wanted to say that vaping has saved my life&quot;</td>
</tr>
<tr>
<td>Negative</td>
<td>Twitter</td>
<td>&quot;When girls post a photo but have a cigarette in their hand #putoff #notcool&quot;</td>
</tr>
<tr>
<td></td>
<td>Reddit</td>
<td>&quot;Day 16 of quitting, just got rid of my vape&quot;</td>
</tr>
<tr>
<td>Neutral</td>
<td>Twitter</td>
<td>&quot;You are my nicotine, heroin, novocaine. Fvck. You're mah drug. Im addicted to you.&quot;</td>
</tr>
<tr>
<td></td>
<td>Reddit</td>
<td>&quot;Tobacco stocks rise as FDA delays plan to cut nicotine levels in cigarettes&quot;</td>
</tr>
</tbody>
</table>

Further benchmark state-of-the-art text classification models for the supervised and semi-supervised learning tasks of identifying product and sentiment in Twitter and Reddit. To the best of our knowledge, this is the first cross-platform semi-supervised attempt in tobacco research.

## 2 Related Work

There has been considerable work in the field of health and social media. While Salathé and Kandelwal (2011) explored Spatio-temporal sentiment towards the new influenza vaccine in Twitter, Cole et al. (2016) studied the quality characteristics of information found in online health forums like Reddit, Mumsnet, and Patient covering HIV, diabetes, and chickenpox. Moreover, Park and Conway (2018) tracked public interest in Ebola, e-cigarettes, influenza, and marijuana on Reddit.

Recent academic works have analyzed tobacco-related trends in multiple social media platforms, including Twitter, Reddit, Instagram, and YouTube (Allem et al., 2019; Carroll et al., 2012; Yanamandra et al., 2020; Zhang et al., 2018). While Allem et al. (2018) explored a thematic analysis of hookah Twitter posts, Allem et al. (2017a) explored e-cigarette trends with an emphasis on social bot behaviour. In Kim et al. (2015), an infouvelle study was performed on e-cigarette tweets on the themes of marketing and usage locations. Moreover, Chen et al. (2015) compared consumer experiences across online discussion forums, including Reddit. They focused on hookah, e-cigarette, and cigarettes using topic modelling and visualization. Sharma et al. (2016) performed a qualitative thematic analysis to determine the limitations and motivations for e-cigarette users with mental illness on Reddit.

The analysis of tobacco sentiment analysis is a relatively less-explored problem. In Allem et al. (2017b), the authors conducted the sentiment analysis of hookah-related tweets using SVM. Moreover, Myslín et al. (2013) performed content and sentiment analysis on tobacco-related tweets using Naive Bayes, KNN, and SVM.

## 3 Dataset

We use the SmokPro (Yanamandra et al., 2020) dataset, which consists of 2,116 tobacco-related tweets classified into the following five distinct product classes: traditional tobacco product mention, modern tobacco product mention, general mention of smoking, narcotics & other drug mentions, and ambivalent mentions. The authors consider cigarette, hookah, pipe, cigar, bidis, cigarillo, shisha, and baccy as traditional tobacco products. Moreover, they consider e-cigarette, e-juice, e-hookahs, e-liquid, mods, vape pens, vapes, tank systems, and electronic nicotine delivery systems (ENDS) as modern tobacco products.

![Table 2: List of subreddits used to scrape data for SentiSmoke-Reddit.](image)

Table 3 illustrates the schema used to determine each product’s sentiment. Our guidelines are centered around the type of tobacco product based on the content. We have also considered street terms and colloquial slangs associated with tobacco product usage. We compile a list of 20 subreddits to scrape data that contained tobacco-related content. They include both information and cessation platforms and are listed in Table 2.

Using PRAW\(^7\), we scrape a sample of Reddit posts made in the last year. We use Reddit’s Top filter and only consider posts with more than 10 upvotes to reduce spam and fewer interacted-with posts. While Twitter’s character limit is 280, Reddit’s maximum character limits for the title and the body are 300 and 40,000 characters, respectively. To avoid cross-platform discrepancies, we only consider the post’s title for this task due to its

\(^7\)https://github.com/praw-dev/praw
similar character length compared with the tweets. Moreover, the effect of title in a Reddit post has been previously studied by Horne and Adali (2017). The scraped dataset consists of 10,023 titles.

We divide the dataset into three splits: training, consisting of 9,023 titles, val, consisting of 100 titles, test, consisting of 900 titles. Since we utilize semi-supervised learning for this task, we only annotate the val and test splits to determine the quality of the learning process. We annotate product labels using the SmokPro’s product annotation schema. We then use the aforementioned sentiment schema (Table 3) for annotating sentiment labels. The annotation process was done by two human annotators having a fluent English background. To assess the annotation standard, we calculate the Inter-Annotator Agreement (IAA) using Cohen’s Kappa coefficient (Fleiss and Cohen, 1973). We obtain Kappa Scores of 0.896 for sentiment annotation for the SentiSmoke-Twitter dataset and 0.840, and 0.927 for sentiment and product identification for the SentiSmoke-Reddit dataset, respectively. These indicate the high quality of all three annotation processes.

<table>
<thead>
<tr>
<th>Positive</th>
<th>Negative</th>
<th>Neutral</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Usage</td>
<td>Past Usage Before Quitting</td>
<td>Sarcastic or Unclear Usage</td>
</tr>
<tr>
<td>Positive Experience of the Product</td>
<td>Negative Experience of the Product</td>
<td></td>
</tr>
<tr>
<td>Cravings</td>
<td>Cravings experienced during current/past journey of Quitting</td>
<td></td>
</tr>
<tr>
<td>Depicts the usage by a second-person or third-person</td>
<td>Shows antipathy towards the product and users</td>
<td>General Advertisements</td>
</tr>
<tr>
<td>Advertisements praising the product</td>
<td>Advertisements showcasing negative aspects or rehabilitation products</td>
<td></td>
</tr>
<tr>
<td>Product reviews on various brands and flavours</td>
<td>Studies and statistics showcasing negative aspects and addictive nature</td>
<td>The tobacco product is not the focus of the text.</td>
</tr>
<tr>
<td>Campaigns supporting usage and condemning regulatory steps</td>
<td>Quitting and Cessation Campaigns and Movements</td>
<td>Usage of smoking-related words in other contexts. For example, &quot;smoking gun&quot;, &quot;smoking hot&quot;.</td>
</tr>
<tr>
<td>Complaining about minor inconveniences occurring due to tobacco-use</td>
<td>Indicates that the interlocutor is acting absurdly by comparing to being intoxicated. For instance, &quot;What are you smoking?&quot;</td>
<td>Other products and places named after smoking-related terms. For example, &quot;cigarette pants&quot;, &quot;tobacco docks&quot;.</td>
</tr>
</tbody>
</table>

Table 3: The data annotation schema used for Tobacco Product Sentiment Analysis.

4 Methodology

This section briefly describes the methodology used for our benchmarking experiments and details the semi-supervised text classification for the Reddit-based dataset.

4.1 Models Used

4.1.1 FastText

FastText (Joulin et al., 2016) is an open-source, free, lightweight NLP library. It maintains a memory-efficient mapping of n-grams and shares information across classes through a hidden representation. FastText uses a hashtable for the word and character n-grams, with the hashtable size directly impacting the size of a model. This library’s accuracy has been found on par with deep neural networks while requiring a fractional amount of training time.

4.1.2 BERT

BERT (Devlin et al., 2019) is a contextualized language representation model based on bidirectional transformers. It uses novel pre-training objectives like masked-language modelling and next-sentence prediction, which enhance the modelling of a relationship between two sentences. In masked-language modelling, the model randomly masks a random subset of the input tokens, and the objective is to predict the correct tokens based purely on context. On the other hand, the next-sentence prediction simultaneously pre-trains text-pair representations. These features help BERT outperform previous state-of-the-art techniques by a large margin. It uses word-piece tokenization and embeddings, which splits parts of words to get better word information and decreases overall vocabulary.
size effectively. We benchmark *cased BERT\textsubscript{Base} and *cased BERT\textsubscript{Large}, and fine-tune them for our classification experiments.

### 4.1.3 RoBERTa
RoBERTa (Liu et al., 2019) is a BERT-based model with improved training methods, larger training data size, and higher computational power. It is trained on ten times the training data as BERT. RoBERTa has been used in multiple online NLP studies published in the last few years in areas including disclosure modelling (Dadu et al., 2020).

In the improved training methodology, dynamic masking replaces BERT’s next-sentence prediction. In dynamic masking, masked tokens change in between epochs. RoBERTa uses larger byte-pair encoding (BPE) vocabulary compared to BERT. These changes led RoBERTa to outperform BERT on GLUE benchmarks. We use the large variant of RoBERTa, RoBERTa\textsubscript{Large}, and fine-tune it for our experiments.

### 4.1.4 DistilBERT
DistilBERT is another BERT-based model utilizing knowledge distillation leading to a much smaller and faster model. It uses 40% less parameters than *Uncased BERT\textsubscript{Base}, runs 60% faster and preserves 95% of *Uncased BERT\textsubscript{Base}’s performance measured on GLUE benchmark. Distilled models have been previously used in downstream tasks with a good predictive performance including subjective bias detection (Pant et al., 2020). We fine-tune the pre-trained *cased base variant of the model for our experiments.

### 4.2 Transfer Learning
We exploit pseudo-labelling to use the unannotated data scraped from Reddit to enhance the process of cross-platform transfer from Twitter to Reddit. For this task, we utilize the predictions from the best-performing model for each subtask. We then use the pseudo-labelled corpus and splits of the original Twitter-based datasets to predict labels for the Reddit evaluation split. This semi-supervised learning methodology is used to exploit the learnings from both Reddit and Twitter.

### 5 Experiments
In this section, we describe the text classification experiments performed using both supervised and semi-supervised learning. We also highlight the experimental settings along with the experimental results for each of the three experiments.

We conduct three experiments with the aforementioned datasets:
Sentiment Identification in Twitter is a supervised experiment that entails predicting the text’s sentiment in the manually annotated SentiSmoke-Twitter dataset.

Product Identification in Reddit is a semi-supervised experiment that entails predicting the tobacco product of the text in the SentiSmoke Reddit dataset. We use the cased variant of BERT<sub>Large</sub>, which had the highest predictive performance in the Twitter dataset (Yanamandra et al., 2020), to pseudo-label the dataset as described in Subsection 4.2.

Sentiment Identification in Reddit is another semi-supervised experiment that entails predicting the sentiment of the text in the SentiSmoke-Reddit dataset. We use RoBERT<sub>Large</sub> to pseudo-label the dataset since it performed the best on Twitter.

We evaluate all the models on the following metrics: F1, Precision, Recall, and Accuracy. Moreover, for FastText, we use its automatic hyperparameter optimization functionality and validate it for 100 validation trials. For all BERT-based models and their distilled variants, we use a learning rate of $1 \times 10^{-5}$ with a weight decay of 0.01, and an `adam epsilon` value of $1 \times 10^{-8}$ while fine-tuning the models. We use a maximum sequence length of 100 and fine-tune the models for 2 epochs.

5.1 Results

We observe that RoBERT<sub>Large</sub> outperforms all other models for all metrics for all the three experiments. From Table 4, we see that it obtains a high F1 score of 0.724 for the Sentiment Identification in Twitter (Supervised) task.

As illustrated in Table 5, experimental results show that RoBERT<sub>Large</sub> again outperformed all other models for all metrics getting an F1 score of 0.573 for the Product Identification in Reddit (Semi-Supervised) task. For the Sentiment Identification in Reddit task, we see that RoBERT<sub>Large</sub> outperformed all other models in F1 and Precision, scoring 0.456 and 0.551 on both metrics, respectively. On the other hand, Cased BERT<sub>Large</sub> obtains the highest Accuracy of 49.00% and Recall of 0.490. The performance of the models in the semi-supervised domain shows that the inductive transfer from Twitter to Reddit setting was effective for the tobacco-product-identification and sentiment-identification task.

For all three experiments, we infer that DistilBERT performs competitively with the large variant of its undistilled counterpart while taking the significantly lower time and computation power for the process of pre-training and fine-tuning.

6 Discussion

In the year 2016 alone, an estimated 10.5 million US youth were exposed to e-cigarette advertise-
ments through the internet. CDC and FDA also discourages e-cigarette-related purchases through informal sources like friends or online market-places and forums. Reddit was one of these online forums aforementioned where sales of tobacco products like e-cigarettes happened until the recent policy update prohibiting the sale of controlled substances like guns and drugs. Therefore, we find it necessary to understand tobacco-related discussions and public sentiment to help shape better policies aimed at bio-surveillance and tobacco control measures.

Previous research (Benson et al., 2020; Pant et al., 2019) in sentiment analysis, and topical tobacco research is heavily reliant on manually annotated datasets. These pose several challenges: expensive to make, limited in scope, difficult to modify, and harder to scale. Additionally, previous research in this sphere is mostly concentrated only on a single social media platform. Both Reddit and Twitter have similar user demographics and a comparable number of monthly active users with a vibrant discourse on health-related issues. Our cross-platform supervised learning approach helps solve data scarcity and scalability while leveraging insights and context from one platform to another. We can see similarities of frequently used words like smoking, vaping, weed, cigar in word clouds generated for SentiSmoke-Twitter in Figure 2 and SentiSmoke-Reddit in Figure 3.

Moreover, extracting data from moderated topic subreddits directly instead of general keyword search used in previous studies (Park and Conway, 2017, 2018) helps us access targeted tobacco-related discourse while weeding out spam and un-

We also perform a part-of-speech-based analysis for comparing between the SentiSmoke-Twitter and SentiSmoke-Reddit, illustrated in Table 6. We have used spaCy for this task. We note a high degree of similarity between the two datasets in terms of part-of-speech.

7 Conclusion and Future Works

This work explored sentiment and product identification on tobacco-related text from two social media platforms: Twitter and Reddit. We released two datasets for multiclass classification annotated across two axes: tobacco product and sentiment. We utilized semi-supervised learning on Reddit text using manually annotated text for Twitter. We then perform benchmarking experiments for sentiment and product identification in Reddit and Twitter using commonly used text classification models like FastText, BERT, RoBERTa, and DistilBERT. We obtain F1 scores as high as 0.72 for supervised sentiment identification in Twitter text, using manually annotated data. Our semi-supervised experiments on product and sentiment identification in Reddit text, using features learned from the Twitter text, obtain the F1 scores of 0.57 and 0.46, respectively.

Future work may involve using the predicted information in recommender systems, expanding the tasks for other social media platforms, and exploring the use of metadata-derived information and comments for the task. Our study can also be extended to real-time monitoring and bio-surveillance tools for social media, which takes continuous inflow of unseen data.
References


Jon-Patrick Allem, Emilio Ferrara, Sree Priyanka Uppu, Tess Boley Cruz, and Jennifer B Unger. 2017a. E-cigarette surveillance with social media data: Social bots, emerging topics, and trends. JMIR Public Health Surveill, 3(4).


Tanvi Dadu, Kartikey Pant, and Radhika Mamidi. 2020. Bert-based ensembles for modeling disclosure and support in conversational social media text.


Kartikey Pant, Venkata Himakar Yanamandra, Alok Debnath, and Radhika Mamidi. 2019. Smokeng: Towards fine-grained classification of tobacco-related social media text. In W-NUT@EMNLP.


Venkata Himakar Yanamandra, Kartikey Pant, and R. Mamidi. 2020. Smokpro: Towards tobacco product identification in social media text. In SI-IRH@ECIR.

Improving Evidence Retrieval with Claim-Evidence Entailment

Fan Yang∗
Amazon Alexa
Cambridge, MA, USA
fyaamz@amazon.com

Eduard Dragut
Temple University
Philadelphia, PA, USA
edragut@temple.edu

Arjun Mukherjee
University of Houston
Houston, TX, USA
arjun4787@gmail.com

Abstract
Claim verification is challenging because it requires first to find textual evidence and then apply claim-evidence entailment to verify a claim. Previous works evaluate the entailment step based on the retrieved evidence, whereas we hypothesize that the entailment prediction can provide useful signals for evidence retrieval, in the sense that if a sentence supports or refutes a claim, the sentence must be relevant. We propose a novel model that uses the entailment score to express the relevancy. Our experiments verify that leveraging entailment prediction improves ranking multiple pieces of evidence.

1 Introduction
Claim verification verifies the credibility of a textual claim by inferring relevant and reliable textual evidence. An example in this space is FEVER, which regards Wiki pages as potential evidence and creates claims by crowdsourcing (Thorne et al., 2018). They propose a three-step pipeline: (i) document-level evidence retrieval; (ii) sentence-level evidence retrieval; (iii) claim-evidence entailment. Some works follow the pipeline and propose new models to improve claim verification (Yoneda et al., 2018; Nie et al., 2019; Hanselowski et al., 2018; Zhou et al., 2019), while other works combine the second and the third step and leverage all possible sentences for claim verification (Yin and Roth, 2018; Ma et al., 2019). We refer to the former as the pipeline framework and the latter as the multi-task framework.

The pipeline framework restricts a few sentences, and therefore it may not cover relevant evidence. The multi-task framework includes all possible sentences, where irrelevant ones may bring overwhelming noise and hurt claim verification. We argue that previous works focus on improving individual components but neglect to examine how those components connect. For example, will the entailment improve the retrieval?

We hypothesize that claim-evidence entailment can provide useful signals for evidence retrieval: if a sentence supports or refutes a claim, the sentence must be relevant. As in Table 1, the first candidate (actual evidence) shares more words and longer phrases than the other candidates. In contrast, the other two candidates may be relevant to the claim to some extent: the second sentence mentions Roddick and masters series and the third sentence mentions masters series and 2002. Thus, we propose a novel method to link the entailment prediction to the relevance score. Our method predicts the entailment for all retrieved candidates and utilizes the entailment score to express the relevancy.

To our knowledge, this is the first work that

Table 1: The entailment result can imply whether the candidate is relevant or not.

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Candidate:</strong> Roddick was ranked in the top 10 for nine consecutive years between 2002 and 2010, and won five Masters Series in that period. <strong>Label:</strong> REFUTE</td>
</tr>
<tr>
<td><strong>Candidate:</strong> Roddick’s hard-court record in 2003 included his first Masters Series titles coming at Canada and Cincinnati and his only Grand Slam title. <strong>Label:</strong> NOT ENOUGH INFO</td>
</tr>
<tr>
<td><strong>Candidate:</strong> Federer won his first Master Series event at the 2002 Hamburg Masters on clay, over Marat Safin. <strong>Label:</strong> NOT ENOUGH INFO</td>
</tr>
</tbody>
</table>

The paper was done before the author joined Amazon.com Inc.
uses the entailment prediction to measure relevancy. Our experiment demonstrates that a reliable entailment prediction improves evidence retrieval. This is beyond previous works that merely share low-level text encoder and train the two steps together.

2 Method

We adopt two base models, the Decomposable Attention (DA) model (Parikh et al., 2016) and the Enhanced Sequential Inference (ESI) model (Chen et al., 2017a), to encode claim-evidence pairs. Both models are designed for textual entailment (Dagan et al., 2005; Bowman et al., 2015; Parikh et al., 2016; Williams et al., 2018). Although there are more methods in the area of textual entailment (Sha et al., 2016; Chen et al., 2017b; Conneau et al., 2017; Nie et al., 2019; Munkhdalai and Yu, 2017; Tay et al., 2018; Ghaeini et al., 2018), we prefer the DA model and the ESI model because they have been widely applied for sentence retrieval and claim-evidence entailment (Thorne et al., 2018; Hanelowski et al., 2018; Nie et al., 2019; Yoneda et al., 2018).

2.1 Claim Verification Pipeline

We follow the three-step pipeline as proposed in (Thorne et al., 2018). We first apply the strategy proposed in (Hanelowski et al., 2018) to retrieve documents. It employs the constituency parser from AllenNLP (Gardner et al., 2018) to find entities. It uses MediaWiki API to obtain relevant articles by matching the title of the article with claim entities. Once we collect \( K \) document candidates, we treat each sentence of the article as potential evidence. Evidence retrieval considers the claim and all candidate sentences as the input and outputs evidence by selecting a subset of sentences. We use \( h_i^R = \text{Enc}^R(w_i^c, w_i^e) \) to denote the process that the relevance encoder encodes the claim \( w_i^c \) and the \( i \)-th evidence candidate \( w_i^e \) into the representation \( h_i^R \).

We obtain the relevance score \( s_i \) by giving \( h_i \) to a fully connected network (FCN). After sorting the relevance score of evidence candidates, we collect the top \( K \) candidates as the evidence. Claim-evidence entailment predicts three probable outcomes: (i) the evidence supports the claim; (ii) the evidence refutes the claim; (iii) the evidence needs more information. The entailment encoder encodes the claim \( w^c \) and the retrieved evidence \( w^e \), and we denote the process as \( h^E = \text{Enc}^E(w^c, w^e) \). Then we feed \( h^E \) to another FCN for the entailment probability.

2.2 Sentence Pair Encoder

We design the relevance encoder and the entailment encoder to share the same architecture, because they both take two sentences as input and produce vector representation that captures claim, evidence, and the correlation of them. Although we consider the DA model and the ESI model in this work, we do not limit the choice of architectures. Let \( a \) and \( b \) be two sentences. The core idea of the two models is to obtain the attention weights \( e_{j,k} \) of word \( a_j \) and word \( b_k \) as in equation 1, where \( F(x) \) follows either DA or ESI to encode a single sentence. With the attention weights \( e \), we obtain \( \bar{a} \) and \( \bar{b} \):

\[
e = F(a)^	op F(b), \ e \in \mathbb{R}^{n_a \times n_b} \tag{1}
\]

\[
\bar{a}_j = \sum^K_{k=0} \frac{\exp(e_{j,k})}{\sum^K_{m=0} \exp(e_{j,m})} F(b)_k \tag{2}
\]

\[
\bar{b}_k = \sum^K_{j=0} \frac{\exp(e_{j,k})}{\sum^K_{m=0} \exp(e_{m,k})} F(a)_j \tag{3}
\]

Then DA and ESI introduce \( G(x_1, x_2) \) to update the representation by taking \((F(a), \bar{a}), \) or \((F(b), \bar{b})\), as the input. Formally, \( h^a = G(F(a), \bar{a}) \) and \( h^b = G(F(b), \bar{b}) \). We recommend readers to refer the origin papers for the implementation of \( F(x) \) and \( G(x_1, x_2) \). We concatenate the two representations as the final output of the encoder, \( h = [h^a, h^b] \). We use the encoder for the retrieval step and the entailment step by varying the input pairs.

2.3 Entailment Score as Relevance Measure

A common design of \( V(x) \) is to generate three values \([v^S, v^R, v^N]\), representing the evidence supports the claim, the evidence refutes the claim, and the evidence does not have enough information, respectively. The largest value decides the entailment: \( v = \max([v^S, v^R, v^N]) \). Intuitively, if one sentence supports or refutes the claim, the sentence must be relevant. Thus, we apply \( V(x) \) on all candidate sentences and propose a new form of the relevance score in Equation 4.

Also, we can combine the new relevance score with the old one that intends to capture the relevance on a single sentence. We introduce the final relevance score \( r^{\text{com}} \) in Equation 5, where \( \alpha \) and \( \beta \) can be learnable parameters or fixed hyperparameters, and \( r^{rFCN} \) is the common way that ob-
tains the relevance score via a fully connected network.

\[ r^{v\text{Diff}} = \max([v^S, v^R]) - v^N \quad (4) \]

\[ r^{\text{com}} = \alpha \cdot r^{\text{FCN}} + \beta \cdot r^{v\text{Diff}} \quad (5) \]

We optimize the retrieval objective as in Equation 6, by ranking the minimum score of evidence and the maximum score of irrelevant sentences. We use cross-entropy as in Equation 7 for the entailment objective. We sum the two as the joint training objective: \( \mathcal{L} = \mathcal{L}^R + \mathcal{L}^V \).

\[ \mathcal{L}^R = \frac{1}{Nc} \sum_i^{Nc} \max(0, \min([r^{+}_{i,0} \ldots r^{+}_{i,N+}]) \]
\[ - \max([r^{-}_{i,0} \ldots r^{-}_{i,N-}]) + m) \]

\[ \mathcal{L}^V = -\frac{1}{Nc} \sum_i^{Nc} y_i^c \log(v_i) \quad (7) \]

One may wonder if a claim requires multiple sentences to form evidence. In that case, \( v \) may predict a single sentence irrelevant. We argue it is not a concern because the \( r^{+} \) in our design is capable of taking a negative value while the \( r^{-} \) can take a positive value. As long as \( r^{+} \) is greater than \( r^{-} \), we can retrieve the right evidence.

3 Experiments

The focus of the experiments is to understand if the entailment score can benefit the retrieval. We conducted experiments on the FEVER dataset (Thorne et al., 2018). FEVER contained 80,035 Support claims, 29,775 Refute claims, and 35,639 NotEnoughInfo claims for training. The shared task of FEVER released 6,666 Support claims, 6,666 Refute claims, and 6,666 NotEnoughInfo claims for validation, and held another blind test set of 6,666 Support claims, 6,666 Refute claims, and 6,666 NotEnoughInfo claims. We considered two scenarios in our experiment, and we describe them as follows:

**EC**: short for Entailment Comparison. We explored the claim-evidence entailment by augmenting gold evidence with irrelevant sentences. We varied the irrelevant sentences so that we maintained the recall of the evidence retrieval. This scenario emphasizes the importance of evidence retrieval.

**RC**: short for Retrieval Comparison. We followed the three-step pipeline and focused on sentence retrieval. At the document retrieval step, we adopted the strategy of (Zhou et al., 2019; Hanselowski et al., 2018). At the sentence retrieval step, we trained on gold evidence and applied retrieved documents for validation.

3.1 The Importance of Evidence Retrieval

We first experimented against the EC to understand the importance of evidence retrieval. The irrelevant sentences are sampled from the same document as the evidence. For claims that did not have gold evidence, we sampled sentences from high ranked documents. We evaluated cases where the evidence contained [5, 10, 15, 20, 25, 30] sentences, while we constructed evidence with [5, 15, 25] sentences for training. Besides, we included the oracle setting that claims were paired with only gold evidence.

We report the result in Figure 1. We notice a clear trend that having irrelevant sentences hurt claim verification, which strengthens the importance of evidence retrieval. We also see that the ESI model performs better than the DA model in all cases, possibly because the ESI model leveraged sequential orders.

3.2 Evidence Retrieval

We conducted experiments against the RC scenario to investigate if the claim-evidence entailment can enhance evidence retrieval. We considered three variants of the sentence retrieval step for comparison: \( R \) was the baseline that no entailment signal was used; \( R+V-J \) measured the relevance score as in Equation 5; \( R-J \) also leveraged the entailment task but only used \( r^{v\text{FCN}} \) for the relevance score.

We selected three previous works to compare against: **TwoW** (Yin and Roth, 2018) combined the retrieval step and the entailment step as **R-J**.
We evaluated the retrieval step on three metrics: **F-recall** is the FEVER recall that measures if the top five sentences contained evidence. F-recall would count true positive if at least one evidence was found; **MRR** stands for mean reciprocal rank. Not only measuring if one evidence was selected, but it also considers the highest ranking position of the evidence. MRR favors to select one evidence as confident as possible; **MAP** stands for mean average precision. It cares for all evidence to be highly ranked and encourages the retrieval step to have all evidence confidently selected so that the retrieved candidates had less irrelevant sentences.

We report the result in Table 2. Since GEAR only reported results on the top five sentences, we calculated MRR and MAP on top five sentences (MRR@5 and MAP@5) and all sentences (MRR@A and MAP@A). We first observe that leveraging the entailment signal improves evidence retrieval on the ESI model, whereas it shows no improvement in the DA model. One possible reason is that the DA model did not perform well on claim verification compared to the ESI model. Therefore, the DA model could not provide a reliable entailment signal to enhance the retrieval. The ESI model, showing better accuracy to predict the entailment, improves MAP and MRR when we leveraged the entailment prediction (ESI-R+V-J v.s. ESI-R), which reinforced the thought that leveraging the entailment signal would require a good entailment predictor.

We also observe that TwoW and HAN could not efficiently retrieve relevant evidence as other methods. Although they show descent accuracy on claim-evidence entailment, a low F-recall means that filtering out low-rank candidates removed relevant evidence as well. Thus, these models show a disadvantage when people care about the evidence that leaded to a verification result.

Finally, we observe that leveraging the entailment signal did not offer an improvement in terms of F-recall. This might indicate that our method benefits ranking multiple pieces of evidence, as we see better performance on MAP and MRR. Besides, GEAR deployed an ensemble of ten models for retrieval, which could explain the difference.

<table>
<thead>
<tr>
<th></th>
<th>TwoW</th>
<th>HAN</th>
<th>GEAR</th>
<th>DA</th>
<th>ESI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>F-Recall</strong></td>
<td>54.81</td>
<td>53.60</td>
<td><strong>86.72</strong></td>
<td>85.59</td>
<td>85.51</td>
</tr>
<tr>
<td><strong>MRR@5</strong></td>
<td>-</td>
<td>-</td>
<td>85.19</td>
<td><strong>82.72</strong></td>
<td>82.69</td>
</tr>
<tr>
<td><strong>MAP@5</strong></td>
<td>-</td>
<td>-</td>
<td>84.10</td>
<td><strong>81.73</strong></td>
<td>81.29</td>
</tr>
<tr>
<td><strong>MRR@A</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td><strong>83.16</strong></td>
<td>83.14</td>
</tr>
<tr>
<td><strong>MAP@A</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td><strong>80.04</strong></td>
<td>80.19</td>
</tr>
</tbody>
</table>

Table 2: Evidence Retrieval Comparison. R is the retrieval baseline. R-J leverages the entailment signal via joint training. R+V-J measures the relevance score as in Equation 5. We use T-test, and **∗∗∗**, **∗∗**, and **∗** means the difference was significant under level $\alpha < 0.01$, $\alpha < 0.05$, and $\alpha < 0.1$, respectively.

![Figure 2: The visualization of the equation 1 using ESI. Each row corresponds to a token of the claim, and each column is a token of the evidence. A darker color means a higher attention score.](image-url)
3.3 Visualization

In Figure 2 we provide one visualization of the claim-evidence attention. We see that the claim and the evidence are attending on the same words and phrases. This explains why the entailment can benefit the retrieval: they reinforce each other to find similar lexicons.

4 Conclusion

In this work, we show that leveraging the entailment prediction can improve evidence retrieval when the entailment step produces a reliable result. In the future, we will adopt pre-trained models, e.g., BERT (Devlin et al., 2019), for our experiments. We expect improvement because BERT shows competitive results on the textual entailment tasks (Zhou et al., 2019).

Ethical consideration This work conducts experiments on benchmark datasets that have been extensively studied in the literature. Although the datasets used in the work was manually annotated, there is no identity characteristics. Also, we use RNN-based models with only a few layers, which are more eco-friendly compared to transformer based models.

Acknowledgments

Research was supported in part by grants NSF 1838147, NSF 1838145, ARO W911NF-20-1-0254. The views and conclusions contained in this document are those of the authors and not of the sponsors. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References


Sentence Structure and Word Relationship Modeling for Emphasis Selection

Haoran Yang, Wai Lam
The Chinese University of Hong Kong
{hryang, wlam}@se.cuhk.edu.hk

Abstract

Emphasis Selection is a newly proposed task which focuses on choosing words for emphasis in short sentences. Traditional methods only consider the sequence information of a sentence while ignoring the rich sentence structure and word relationship information. In this paper, we propose a new framework that considers sentence structure via a sentence structure graph and word relationship via a word similarity graph. The sentence structure graph is derived from the parse tree of a sentence. The word similarity graph allows nodes to share information with their neighbors since we argue that in emphasis selection, similar words are more likely to be emphasized together. Graph neural networks are employed to learn the representation of each node of these two graphs. Experimental results demonstrate that our framework can achieve superior performance.

1 Introduction

Emphasis Selection recently proposed by (Shirani et al., 2019) aims to select candidate words for emphasis in short sentences. By emphasizing words, people’s intent can be better conveyed, which is useful in a variety of applications. For example, it can be used in spoken language processing to generate more expressive sentences and be used to enable automated design assistance in authoring, i.e., labeling important parts in a paragraph or in a poster title. Although it seems that this task is highly similar to the task of keyword extraction (Gupta, 2017), there are two different fundamentals. The first difference is that emphasis selection focuses on a paragraph which is composed of multiple sentences while emphasis selection aims to choose words from a short sentence. This difference implies that modeling sentence structure is more effective in emphasis selection. The second difference is that many global word statistics methods employed in keyword extraction such as TF-IDF and word co-occurrence frequency will not work in this task, because for short sentences, it is meaningless to count word frequency and whether the word should be emphasized has nothing to do with the frequency of the word. In addition, keyword extraction requires that the collected keywords are diverse, which means that if two words have similar meaning, only one should be kept. However, in emphasis selection, similar words tend to be emphasized together. Emphasis selection also shares some similarities with entity recognition (Yadav and Bethard, 2018). But one major difference is that the parts of speech of emphasized words are more diverse and the relation of adjacent words is weaker in the emphasis selection task.

Figure 1: Two examples of the emphasis selection. Words with darker background indicate that more people agree to emphasize.

Generally speaking, emphasis selection can be modeled as a sequence classification task where the input is a sentence and the output is each word’s probability to be emphasized. Shirani et al. (Shirani et al., 2019) propose a model which is based on the Recurrent Neural Network (Mikolov et al., 2010) and KL-Divergence loss function. Despite the fact that it looks like a straightforward task, there still exist some challenges. The first challenge is about how to incorporate sentence structure information into the model. Sentence structure information includes what role (subject, predicate, object, etc.) the word plays as well as the position of the word in a sentence. Obviously, this kind of information is very useful. Existing works (Shi-
rani et al., 2019) fail to model the global structure of a sentence. The second challenge is that there is no given context except a short sentence, so it requires the model to be able to capture some common patterns or regularities of most people. More concretely, if two words are similar, they are more likely to be emphasized together. For example, in Figure 1, persistence and victory are more likely to be emphasized together. This observation can also be found in the second example: Never and impossible. Moreover, we analyze the training dataset and get a more concrete understanding of this phenomenon through the following procedures: For each training sentence, we consider the most popular emphasized word called word A. Then, we identify the most similar word called word B to the word A based on GloVe embedding (Pennington et al., 2014). We find that the word B is also emphasized with a higher probability than other words in this sentence and this phenomenon occurs in about 26% of the training dataset. Therefore, modeling this kind of relationship between words definitely can help improve the performance of models.

In this paper, we propose a sentence structure graph to handle the sentence structure issue. Specifically, the sentence structure graph is derived from the parse tree of a sentence which contains useful information for this task. For example, as illustrated in Figure 2, when the path is S→NP→PRP, the word I is not inclined to be emphasized since this path indicates that this word is a subject. However, when the path is S→VP→S→VP→NP→NN, the word basketball is likely to be emphasized since this word is a noun in a verb phrase. Generally, such sentence structure graph can reveal the role of words in a sentence which is beneficial for emphasis selection. Another important information - word relationship information is captured by a word similarity graph. Through the word similarity graph, words can share information with their neighbours, resulting in similar emphasized probabilities of similar words. Next, graph neural networks (Vaswani et al., 2017; Cai and Lam, 2020; Kipf and Welling, 2017; Wu et al., 2019; Yun et al., 2019; Veličković et al., 2018) which have been demonstrated effective in modeling graph structure data are employed to learn the representation of each node of these two graphs. We conduct extensive experiments based on different word embeddings, i.e., GloVe (Pennington et al., 2014), ELMo (Peters et al., 2018), RoBERTa (Liu et al., 2019) and the experimental results show that our model can achieve superior performance.

2 Related Work

Emphasis selection is a new task proposed by (Shirani et al., 2019) which aims to choose a subset of words to emphasize in a sentence. Shirani et al. (Shirani et al., 2019) propose a model which is based on the Recurrent Neural Network (Mikolov et al., 2010). KL-Divergence loss function is adopted to conduct the label distribution learning (LDL) (Geng and Zhao, 2014). This method achieves competitive performance over the sequence labeling model: CRF (Lafferty et al., 2001).

In Recent years, graph neural networks (Wu et al., 2019; Kipf and Welling, 2017; Yun et al., 2019; Veličković et al., 2018; Cai and Lam, 2020) have demonstrated superiority in modeling the structure of graphs. Kipf et al. (Kipf and Welling, 2017) propose a graph convolutional network which is based on the fourier theory. One drawback of this model is that the edge weight of the graph needs to be known in advance. To overcome this shortcoming, Petar et al. (Veličković et al., 2018) use a masked self-attention layer to calculate the weight of node’s neighbours dynamically and then aggregate information by conducting a weighted addition operation. Currently, graph neural networks are applied to various tasks. Feria et al. (Feria et al., 2018) construct a word graph by calculating the word embedding similarity and apply the community detection algorithm to find different communities. Through the graph, they can find named entities for a bilingual language base in an
unsupervised manner. Sun et al. (Sun et al., 2019) put forward a diverse graph pointer network for keyword extraction. They first construct a word graph based on the distance of two words and then use the graph convolutional network as an encoder to obtain each node’s representation, finally a pointer network decoder and the diverse mechanism are employed to generate diverse keywords. The graph encoder can capture document-level word salience and overcome the long-range dependency problem of RNN.

3 Methodology

We follow the same problem setting given by (Shirani et al., 2019). Suppose a sentence is composed of $n$ words $C = (x_1, x_2, ..., x_n)$. Our goal is to obtain a subset $S$ of words in $C$ as selected words for emphasis where $1 \leq |S| \leq n$.

We model this task as a prediction problem:

$$(p_1, p_2, ..., p_n) = \text{model}(x_1, x_2, ..., x_n) \quad (1)$$

where $p_i$ is $i$-th word’s probability to be emphasized. Then $S$ contains the top-$|S|$ words with high probability.

Figure 3 depicts the architecture of our proposed model which is composed of three parts: (i) the middle part - sequence encoder (ii) the left part - word similarity graph encoder (iii) the right part - sentence structure graph encoder. Next, we will provide a detailed description of each part.

3.1 Sequence Encoder

The sequence encoder is composed of an embedding layer and a bidirectional GRU. It is mainly used to model the sequence information, i.e., word sequence and tag sequence. Formally, given a sentence $C = (x_1, x_2, ..., x_n)$ with $n$ words, the embedding layer is responsible for converting each word into a $d_1$-dimensional vector and converting the corresponding POS tag into a $d_2$-dimensional vector:

$$(w_1, ..., w_n) = \text{WordEmbed}(x_1, ..., x_n) \quad (2)$$

$$(e_1, ..., e_n) = \text{TagEmbed}(t_1, ..., t_n) \quad (3)$$

where $(t_1, ..., t_n)$ is the POS tag sequence and $w_i \in \mathbb{R}^{d_1}, e_i \in \mathbb{R}^{d_2}$. Then the word embedding and the tag embedding are concatenated and fed into a encoder $E$ to encode the sequence information.

3.2 Word Relationship Modeling

Given a sentence, we take each word as a node and the weight of the edge is calculated by the word embedding similarity. The weight matrix is denoted by $A \in \mathbb{R}^{n \times n}$. After the graph is constructed, a $L$-layer graph convolutional network (GCN) (Kipf and Welling, 2017) is employed to encode the word similarity graph:

$$H^{l+1} = \text{ReLU}(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}H^{l}W^{l}) \quad (5)$$

where $W^l$ is a parameter and $H^l$ denotes the nodes’ representation in the $l$-th layer. $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix and $D_{ii} = \sum_j A_{ij}$.

Recall that the WSG is a complete graph since each two words are connected by a weighted edge. There exists a serious problem: Useful information may be overwhelmed by useless information, because a majority number of words do not need to be emphasized, causing the information in words that are not emphasized dominates the words that should be emphasized. To alleviate this problem, we adopt two strategies: residual module (He et al., 2016) and gate mechanism (Gehring et al., 2017; Dauphin et al., 2017). The residual module makes the current node’s representation as the addition
between the former representation and the aggregated information from its neighbours. The gate mechanism controls the magnitude of the aggregated information. Through this way, the current node’s representation will not be significantly affected by its neighbours. Therefore Equation (5) can be rewritten as:

\[ M^{l+1} = H^l W^l \] (6)

\[ C = s(H^l W^g) \] (7)

\[ H^{l+1} = M^{l+1} + D^{-\frac{1}{2}} AD^{-\frac{1}{2}} M^{l+1} \otimes C \] (8)

where \( s(\cdot) \) is the sigmoid function and \( \otimes \) is the point-wise multiplication.

We obtain \( H^0 \) from the word embedding matrix and obtain the \( L \)-th layer output \( H^L = (w^L_1, ..., w^L_n) \) as each node’s features of the word similarity graph.

### 3.3 Sentence Structure Modeling

SSG is constructed by parsing the sentence using NLTK\(^1\) and StandfordNLP\(^2\). Then, we remove the leaf nodes (which are the words) and the remaining part is the SSG. Each node of the graph is a kind of POS tag and the path from the root to a specific word can reveal what role the word plays in the sentence.

Apparently, the weight of edges is important. For example, in Figure 2, the root node S has two children nodes NP and VP. The edge (S, NP) should have a smaller weight than the edge (S, VP) since people tend not to emphasize the subject in most circumstances. Different from WSG where the weight can be calculated by the word embedding similarity explicitly, it is not appropriate to calculate the weight in the SSG by the node similarity. Hence, we integrate the idea of Transformer (Vaswani et al., 2017; Cai and Lam, 2020) and masked self-attention (Veličković et al., 2018) to the SSG modeling. Firstly, we generate three vectors: key, query, value, according to the current node’s representation:

\[ k^{l+1}_i = W^k_i (v^l_i) \] (9)

\[ q^{l+1}_i = W^q_i (v^l_i) \] (10)

\[ v^{l+1}_i = W^v_i (v^l_i) \] (11)

where \( W^k_i, W^q_i, W^v_i \) are parameters. \( k^l_i, q^l_i, v^l_i \) correspond to the \( l \)-th layer key, query, value vector respectively. \( v^l_i \) is initialized from the tag embedding matrix. Then, a masked self-attention is employed to allow nodes aggregating information only from their neighbours.

\[ v^{l+1}_i = \sum_{j \in \mathcal{N}(i)} a_{ij} v^{l+1}_j \] (12)

\[ a_{ij} = \frac{\exp(q^{l+1}_i k^{l+1}_j)}{\sum_{z \in \mathcal{N}(i)} \exp(q^{l+1}_i k^{l+1}_z)} \] (13)

where \( \mathcal{N}(i) \) is the neighbour set of the node \( i \). After the graph is encoded with a \( L \)-layer network, we obtain the leaf nodes (the green nodes shown in Figure 2) representation \( V = (v^L_1, v^L_2, ..., v^L_n) \).

### 3.4 Loss Function

After obtaining these three modules’ output, we conduct a concatenation operation and calculate the probability:

\[ p_i = \text{softmax}(f([h_i, v^L_i, w^L_i])) \] (14)

where \( p_i \in \mathbb{R}^3 \) is \( i \)-th word’s probability distribution. \( f \) represents a fully connected neural network. We adopt negative log likelihood as the loss function:

\[ L = - \sum_{C \in \mathcal{D}_{train}} \sum_{i=1}^{\left| C \right|} \log p_{iy_i} \] (15)

### 4 Experiment and Results

#### 4.1 Dataset

We use the dataset\(^3\) provided by (Shirani et al., 2019). The dataset contains 2742 training sentences and 392 test sentences. Each sentence is

\[ \text{Table 1: An example of the labeled dataset} \]

```
<table>
<thead>
<tr>
<th>DIY</th>
<th>ideas</th>
<th>for</th>
<th>leaving</th>
<th>up</th>
<th>your</th>
<th>home</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>B</td>
<td>I</td>
</tr>
<tr>
<td>B</td>
<td>I</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>O</td>
<td>B</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>O</td>
<td>O</td>
<td>B</td>
<td>O</td>
<td>O</td>
<td>B</td>
<td>I</td>
</tr>
<tr>
<td>O</td>
<td>O</td>
<td>B</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>B</td>
<td>O</td>
<td>O</td>
<td>B</td>
<td>O</td>
<td>B</td>
<td>O</td>
</tr>
<tr>
<td>B</td>
<td>I</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>
```
<table>
<thead>
<tr>
<th>Methods</th>
<th>Match-1</th>
<th>Match-2</th>
<th>Match-3</th>
<th>Match-4</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>GloVe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN</td>
<td>0.541</td>
<td>0.678</td>
<td>0.754</td>
<td>0.805</td>
<td>0.695</td>
</tr>
<tr>
<td>RNN (Shirani et al., 2019)</td>
<td>0.536</td>
<td><strong>0.712</strong></td>
<td>0.777</td>
<td>0.811</td>
<td>0.709</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>0.569</strong></td>
<td>0.703</td>
<td>0.772</td>
<td><strong>0.813</strong></td>
<td>0.714</td>
</tr>
<tr>
<td>Ours w/o WSG</td>
<td>0.563</td>
<td>0.710</td>
<td><strong>0.778</strong></td>
<td>0.810</td>
<td><strong>0.715</strong></td>
</tr>
<tr>
<td>Ours w/o SSG</td>
<td>0.561</td>
<td>0.710</td>
<td>0.769</td>
<td>0.811</td>
<td>0.713</td>
</tr>
<tr>
<td>ELMo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN</td>
<td>0.574</td>
<td>0.729</td>
<td>0.795</td>
<td>0.832</td>
<td>0.733</td>
</tr>
<tr>
<td>RNN-based (Shirani et al., 2019)</td>
<td>0.592</td>
<td>0.752</td>
<td>0.804</td>
<td>0.822</td>
<td>0.743</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>0.610</strong></td>
<td><strong>0.768</strong></td>
<td><strong>0.813</strong></td>
<td><strong>0.836</strong></td>
<td><strong>0.757</strong></td>
</tr>
<tr>
<td>Ours w/o WSG</td>
<td>0.604</td>
<td>0.742</td>
<td>0.804</td>
<td>0.827</td>
<td>0.744</td>
</tr>
<tr>
<td>Ours w/o SSG</td>
<td>0.597</td>
<td>0.753</td>
<td>0.801</td>
<td>0.836</td>
<td>0.747</td>
</tr>
</tbody>
</table>

Table 2: Results of our model and baselines on GloVe and ELMo. The best performance is boldfaced.

Table 3: A sample case. Numbers outside the brackets indicate the word’s probability of being emphasized. Numbers in the brackets are the ranking of the corresponding word. (a/b) means that two words have the same ranking.

labeled by nine annotators. Table 1 gives a sample record of one sentence. B, I, O represent the beginning word to be emphasized, the interior word to be emphasized, and the word not to be emphasized respectively. Since there exists different opinions about whether the word should be emphasized, the labels given by nine annotators are slightly different.

4.2 Experimental Setup

We regard each annotator’s labeling as a sample in the dataset. In other words, each sentence is associate with nine samples. In order to verify the robustness of our model, we conduct experiments on two pre-trained word embeddings: 300-d GloVe (Pennington et al., 2014) and 2048-d ELMo (Peters et al., 2018). For the above two kinds of embeddings, we adopt GRU as the encoder $E$. The GRU hidden state size is 512 and 1024 respectively. The word similarity graph’s node embedding size is 300 and 2048 respectively. The sentence structure graph’s node embedding size is 300 and 512 respectively. Moreover, we initialize the sentence structure graph’s node embedding by training a classifier which only uses the sentence structure graph encoder. We adopt a two-layer bidirectional GRU. The sentence structure graph and the word similarity graph are encoded by a two-layer graph neural network. The batch size is set to 16. The negative slope of the ReLU function is set to 0.2. We use the Adam optimizer and the learning rate is 0.0001. The number of epoch is 100. We also add a dropout layer and the dropout rate is 0.5.

Since generalized pretrained language models such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) are demonstrated effective in a large bunch of downstream tasks, we also report results obtained by fine-tuning the RoBERTa on the emphasis selection dataset. There are two different experimental settings. The first setting is that only the RoBERTa model is used as the encoder $E$. The second setting is that a GRU layer is added on the top of the RoBERTa model, i.e., RoBERTa+GRU is the encoder $E$. The sentence structure model and the word relationship model remain unchanged. Adam optimizer is adopted and the learning rate is set to 1e-5.

4.3 Evaluation Metric

We adopt $\text{Match-m}$ (Shirani et al., 2019) as the evaluation metric which is defined as below:

\begin{align*}
\text{Match-m} & \quad \text{For a sentence } C, \text{ we choose } m \text{ words (denoted by } S_m(C) \text{) with the top-m probability (probability of the label B + probability of the label I) in the ground truth and } m \text{ words (denoted} \\
\end{align*}
Table 4: Results of our model and baselines based on two different architectures, RoBERTa and RoBERTa+GRU. The best performance is boldfaced.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Match-1</th>
<th>Match-2</th>
<th>Match-3</th>
<th>Match-4</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoBERTa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours w/o both</td>
<td>0.635</td>
<td>0.756</td>
<td>0.803</td>
<td>0.832</td>
<td>0.757</td>
</tr>
<tr>
<td>Ours w/o WSG</td>
<td>0.640</td>
<td>0.775</td>
<td>0.793</td>
<td>0.827</td>
<td>0.759</td>
</tr>
<tr>
<td>Ours w/o SSG</td>
<td>0.633</td>
<td>0.760</td>
<td>0.804</td>
<td>0.839</td>
<td>0.759</td>
</tr>
<tr>
<td>Ours</td>
<td>0.633</td>
<td>0.779</td>
<td>0.803</td>
<td>0.833</td>
<td>0.762</td>
</tr>
</tbody>
</table>

| RoBERTa+GRU   |         |         |         |         |         |
| Ours w/o both | 0.607   | 0.755   | 0.795   | 0.822   | 0.745   |
| Ours w/o WSG  | 0.602   | 0.766   | 0.798   | 0.825   | 0.748   |
| Ours w/o SSG  | 0.607   | 0.758   | 0.801   | 0.837   | 0.747   |
| Ours          | 0.600   | 0.761   | 0.806   | 0.838   | 0.751   |

Table 5: A failed case. Numbers are the word’s probability of being emphasized.

<table>
<thead>
<tr>
<th>Annotator</th>
<th>Thanks</th>
<th>for</th>
<th>showing</th>
<th>me</th>
<th>all</th>
<th>the</th>
<th>best</th>
<th>dance</th>
<th>moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>0.412</td>
<td>0.057</td>
<td>0.332</td>
<td>0.092</td>
<td>0.063</td>
<td>0.024</td>
<td>0.406</td>
<td>0.599</td>
<td>0.387</td>
</tr>
<tr>
<td>Annotator</td>
<td>0.444</td>
<td>0.111</td>
<td>0.111</td>
<td>0.111</td>
<td>0.111</td>
<td>0.888</td>
<td>0.555</td>
<td>0.444</td>
<td></td>
</tr>
</tbody>
</table>

Experimental results based on RoBERTa are listed in Table 4. Compared with the results based on GloVe and ELMo, RoBERTa and its variants achieve higher average match score which shows that a better initialized word embedding is helpful for a better performance. For the same RoBERTa encoder, Ours can obtain the highest score on Average and Match-2. For RoBERTa+GRU encoder, Ours can obtain the highest score on Average, Match-3 and Match-4. However, one interesting finding is that RoBERTa encoder performs much better than RoBERTa+GRU encoder. Two possible reasons may interpret this phenomenon. The first reason is the overfitting problem and the second reason is that the larger network is harder to train due to some optimization issues, e.g., gradient vanishing.

4.4 Results and Analysis

4.4.1 Experimental Results

We compare our model with the existing model based on RNN proposed by Shirani et al. (2019) and the convolutional neural network (CNN). We report results evaluated by the metrics Match-1, Match-2, Match-3, Match-4 and the average of these four metrics. From Table 2, we can see that CNN lags behind other models on the whole.

When the word embedding is GloVe, models with at least one graph surpass RNN on almost all the metrics except Match-2. In particular, our model can achieve an improvement on Match-1 and Match-4. Our model without WSG (word similarity graph) achieves an excellent performance on Match-3 and Average. When the word embedding is ELMo, ours is superior to RNN-based on all the evaluation metrics. Compared to these two ablated models, Ours can also achieve better performance. Ours w/o WSG is better than RNN-based on all the evaluation metrics except Match-2 and Ours w/o SSG is better than RNN-based except Match-3. On the whole, models with graphs can obtain better results on most metrics compared to the baseline models, which shows the advantage of these two components.

Experimental results based on RoBERTa are listed in Table 4. Compared with the results based on GloVe and ELMo, RoBERTa and its variants achieve higher average match score which shows that a better initialized word embedding is helpful for a better performance. For the same RoBERTa encoder, Ours can obtain the highest score on Average and Match-2. For RoBERTa+GRU encoder, Ours can obtain the highest score on Average, Match-3 and Match-4. However, one interesting finding is that RoBERTa encoder performs much better than RoBERTa+GRU encoder. Two possible reasons may interpret this phenomenon. The first reason is the overfitting problem and the second reason is that the larger network is harder to train due to some optimization issues, e.g., gradient vanishing.

4.4.2 Case Study

To gain some insights of our proposed model, we present a sample case generated by the ELMo-based model as shown in Table 3. We can see that Ours not only predicts the ranking accurately, but also obtains very close probability to the ground truth probability derived by annotators. Besides that, the probabilities of foolish and sane predicted by our model are very close than that predicted by RNN-based, which shows that the word similarity graph can impel similar words to have similar probabilities.
We also provide a failed case in Table 5. It is intrinsically harder to rank the words in this sentence even for human beings. Our model does not rank them correctly on these cases where multiple words may be emphasized.

4.4.3 Some Useful Tips

We conclude some tips on the experiment that leads to better performance. (1) We can firstly train a classifier only using the SSG, then use the pre-trained embeddings as an initialization of the sentence graph nodes embeddings. It can obtain higher score and faster convergence of the model. (2) We also consider another method to model the relationships between words using a self-attention operation proposed by Lin et al. (2017) above the hidden vectors of RNN. However, the performance is slightly degraded compared to removing this operation. So we think it is much better to model words relationships and sequence information separately.

5 Conclusions

The sentence structure graph and the word similarity graph are proposed to solve two issues found in emphasis selection. The sentence structure graph helps to model the structure information of the sentence and the word similarity graph is useful in modeling relationships between words. With the development of graph neural network, the two graphs can be properly encoded and integrated into existing models. Experimental results demonstrate that our framework can achieve superior performance.

References


Miguel Feria, Juan Paolo Balbin, and Francis Michael Bautista. 2018. Constructing a word similarity graph from vector based word representation for named entity recognition.


Xin Geng and Quan Zhao. 2014. Label distribution learning.


Abstract

This study proposes an utterance position-aware approach for a neural network-based dialogue act recognition (DAR) model, which incorporates positional encoding for utterance’s absolute or relative position. The proposed approach is inspired by the observation that some dialogue acts have tendencies of occurrence positions. The evaluations on the Switchboard corpus show that the proposed positional encoding of utterances statistically significantly improves the performance of DAR.

1 Introduction

The recognition of dialogue acts (DAs), which represent the intention or function of each utterance in a dialogue, is useful for various dialogue applications, such as dialogue systems and dialogue summarization. Recently, neural network- (NN-)based approaches have become dominant in dialogue act recognition (DAR) because NN-based models obtained higher performance than other approaches. Most existing NN-based DAR models treat DAR as a sequence labeling problem, where utterances are first encoded with a hierarchical recurrent neural network (RNN), and subsequently, the sequence of DAR labels is identified from the encoded representations using the Conditional Random Field (CRF) (Kumar et al., 2018; Chen et al., 2018; Li et al., 2019; Raheja and Tetreault, 2019). These CRF-based models can capture the local dependencies of DA sequences; however, they cannot model global dependencies due to the first-order Markov assumption. To alleviate this problem, Colombo et al. (2020) proposed a sequence-to-sequence (seq2seq) architecture for DAR, which could learn global dependencies, and achieve the state-of-the-art performance. We focus on this work and employ it as the basis of our proposed model.

Existing NN-based DAR models focus on the context of a dialogue and dependencies between DAs but do not focus on the position of an utterance. However, we have found that some DAs have tendencies of occurrence positions. For example, the “Open-Question” DA in the Switchboard corpus tends to appear at the beginning of a telephonic conversation.

Inspired by this observation, this study proposes an utterance position-aware approach for a NN-based DAR model, which explicitly encodes the position of an utterance by positional encoding and generates DA label sequences based on the hidden vectors augmented with the positional encoding of utterance. In particular, we implement two types of positional encodings of utterances: (1) sinusoidal positional encoding used in Transformer neural machine translation (Vaswani et al., 2017), which represents the absolute position of each utterance; (2) length-ratio positional encoding, which is a positional encoding based on the ratio of the position of an utterance to the dialogue length (i.e., the total number of utterances in the dialogue). The second encoding aims to encode relative positional information to alleviate the variation of dialogue length.

The evaluations on the Switchboard corpus (Stolcke et al., 2000), which is one of the most popular benchmark datasets in DAR, show that the performance of DAR is statistically significantly improved by incorporating the proposed positional encoding of utterances (up to +0.31 precision). Our analysis demonstrates that the proposed model statistically significantly improves the performance of long dialogues.

2 Proposed DAR Model

Figure 1 shows an overview of the proposed model. The proposed model incorporates positional encoding of utterances (Section 2.2) into the baseline seq2seq DAR model (Section 2.1).
2.1 Baseline DAR Model

Our DAR model takes a dialogue document $D = (U_1, \cdots, U_{|D|})$, which is the sequence of utterances $U_i$ as an input and predicts the sequence of DA labels $Y = (\hat{y}_1, \cdots, \hat{y}_{|D|})$ from $D$, where $y_i$ is the DA label of $U_i$ and each utterance is the sequence of words (i.e., $U_i = (w_{i1}, \cdots, w_{i|U_i|})$).

The baseline model is a seq2seq model that consists of a hierarchical encoder and a decoder with guided attention proposed by Colombo et al. (2020). The hierarchical encoder hierarchically encodes a dialogue document using two types of RNN layers, a word layer and an utterance layer. Particularly, the word layer first generates the sequence of intermediate word vectors $(u_{i1}, \cdots, u_{i|U_i|})$ of each utterance $U_i$, from the sequence of word embedding vectors $(w_{i1}, \cdots, w_{i|U_i|})$. Subsequently, the utterance layer generates the sequence of intermediate utterance vectors $(h_{1enc}, \cdots, h_{|D|enc})$ from the outputs of the word layer. We use bi-directional gated recurrent unit (Cho et al., 2014) as RNN. The computations in the encoder are as follows:

$$
  u_{ij} = \text{BiGRU}_{\text{word}}(u_{ij-1}, w_{ij}),
$$

(1)

$$
  h_{ienc} = \text{BiGRU}_{\text{att}}(h_{i-1enc}, u_{i|U_i|}).
$$

(2)

The decoder autoregressively generates the sequence of DA labels after receiving the previous hidden state and the previous output as inputs. In each timestep $i$, the previous label $\hat{y}_{i-1}$ is first converted into an embedding vector $f_{\text{embed}}(\hat{y}_{i-1})$ in the same way as word embedding, following which the hidden vector $h_{i}^{dec}$ is generated as follows:

$$
  e_i = \begin{cases} f_{\text{embed}}(\langle \text{SOS} \rangle) & (i = 1) \\ f_{\text{embed}}(\hat{y}_{i-1}) & (\text{otherwise}), \end{cases}
$$

(3)

$$
  h_{i}^{dec} = \begin{cases} \text{GRU}(h_{i|D|enc}, e_i) & (i = 1) \\ \text{GRU}(h_{i-1}^{dec}, e_i) & (\text{otherwise}), \end{cases}
$$

(4)

where $h_{i}^{enc}$ denotes the encoder’s final hidden state and $\langle \text{SOS} \rangle$ denotes the special label. Finally, the $i$-th DA label is predicted by applying hard guided attention (Colombo et al., 2020), which attends only to the corresponding encoder’s hidden state, as follows:

$$
  z_i = [h_i^{dec}, h_i^{enc}],
$$

(5)

$$
  \hat{y}_i = \text{LogSoftmax}(\text{ReLU}(Wz_i)),
$$

(6)

where $[;]$ indicates a concatenation operation, and $W$ denotes a parameter matrix.

2.2 Positional Encodings of Utterances

The proposed model incorporates positional encoding of utterances into the baseline DAR model described in Section 2.1 to explicitly consider the position of an utterance in the inference of its DA label. Specifically, the proposed model predicts the $i$-th DA label from the following $z_i$, which is the concatenation of the original one and the positional encoding of the $i$-th utterance $h_i^{pos}$ rather than Equation (5).

$$
  z_i = [h_i^{dec}; h_i^{enc}; h_i^{pos}].
$$

(7)

In this work, we implement two types of positional encodings of utterances as $h_i^{pos}$ in Equation (7): (1) absolute positional encoding (PE$_{abs}$), which encodes the absolute positional information of each utterance, and (2) relative positional encoding (PE$_{rel}$), which encodes the relative positional information of each utterance considering the dialogue length.

**Absolute Positional Encoding (PE$_{abs}$)**

We incorporate the absolute positional information of utterances by applying sinusoidal positional encoding used in the Transformer (Vaswani et al.,
Table 1: Statistics of our experiment data.

<table>
<thead>
<tr>
<th>Hyperparameters</th>
<th>Search Range</th>
<th>Optimized Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word embedding size</td>
<td>16 - 512</td>
<td>s2s</td>
</tr>
<tr>
<td>Encoder GRU size</td>
<td>16 - 512</td>
<td>64</td>
</tr>
<tr>
<td>Decoder GRU size</td>
<td>16 - 512</td>
<td>300</td>
</tr>
<tr>
<td>PE size</td>
<td>16 - 512</td>
<td>64</td>
</tr>
<tr>
<td>Clip gradient value</td>
<td>1.0 - 5.0</td>
<td>-</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.1 - 0.7</td>
<td>3.0</td>
</tr>
<tr>
<td>Learning rate</td>
<td>$5 \times 10^{-6} - 1 \times 10^{-2}$</td>
<td>$1 \times 10^{-3}$</td>
</tr>
<tr>
<td>Weight decay</td>
<td>$5 \times 10^{-6} - 1 \times 10^{-2}$</td>
<td>$1 \times 10^{-5}$</td>
</tr>
</tbody>
</table>

Note that our PE_{rel} is the same formulation as the one of Takase and Okazaki (2019). However, Takase and Okazaki (2019) have used the positional encoding for controlling the output length in document summarization. The purpose is different from ours, and our work is the first attempt to introduce a relative positional encoding to DAR.

3 Experiments

3.1 Settings

We evaluated our proposed method on the DAR task with the Switchboard corpus (SwDA) (Stolcke et al., 2000), consisting of 1,155 telephonic conversations using the standard splitting of the corpus (Lee and Dernoncourt, 2016). Table 1 shows the statistics of the SwDA dataset. The corpus comprises 43 different kinds of DA labels, and the vocabulary size is 19K.

We compared our two types of proposed models (s2s+PE_{abs} and s2s+PE_{rel}), each of which incorporates the positional encoding of utterances, described in Section 2.2, into the baseline DAR model (s2s), described in Section 2.1. We measured the performance of DAR as precision. We used Adam optimizer (Kingma and Ba, 2015) to train each DAR model, which is updated using a scheduler with a patience of 20 epochs and a reduced rate of 0.5. We used weight decay, gradient norm clipping, and dropouts (Srivastava et al., 2014). Each model was implemented using PyTorch and trained on a single NVIDIA GeForce GTX 1080 Ti. The hyperparameters of each model were optimized using

2017) to positional encoding of utterances in DAR. Let $d$ be the dimension of the positional encoding. The $k$-th element of the positional encoding of the $i$-th utterance, $PE_{abs}(i,k)$, is calculated as follows:

$$PE_{abs}(i,2k) = \sin \left( \frac{i}{10000^{2k/d}} \right), \quad (8)$$

$$PE_{abs}(i,2k+1) = \cos \left( \frac{i}{10000^{2k/d}} \right). \quad (9)$$

Relative Positional Encoding (PE_{rel})

We incorporate the relative positional information of utterances by length-ratio positional encoding, which encodes the ratio of the position of an utterance to the dialogue length. $PE_{abs}$ could not capture an occurrence phase (e.g., the beginning, middle, or last part) in a dialogue because the same absolute position can appear at different phases. For example, the 10th utterance belongs to the beginning part of a long dialogue (e.g., dialogue length = 100) whereas it belongs to the last part of a short dialogue (e.g., dialogue length = 10). To alleviate the variation of dialogue length, $PE_{rel}$ encodes positional information normalized by the dialogue length $|D|$ as follows:

$$PE_{rel}(i,2k|D|) = \sin \left( \frac{i}{D^{2k/d}} \right), \quad (10)$$

$$PE_{rel}(i,2k+1|D|) = \cos \left( \frac{i}{D^{2k/d}} \right). \quad (11)$$

We compared our two types of proposed models (s2s+PE_{abs} and s2s+PE_{rel}), each of which incorporates the positional encoding of utterances, described in Section 2.2, into the baseline DAR model (s2s), described in Section 2.1. We measured the performance of DAR as precision. We used Adam optimizer (Kingma and Ba, 2015) to train each DAR model, which is updated using a scheduler with a patience of 20 epochs and a reduced rate of 0.5. We used weight decay, gradient norm clipping, and dropouts (Srivastava et al., 2014). Each model was implemented using PyTorch and trained on a single NVIDIA GeForce GTX 1080 Ti. The hyperparameters of each model were optimized using
### Model Precision

<table>
<thead>
<tr>
<th></th>
<th>Baseline Model</th>
<th>Proposed Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>s2s</td>
<td>77.84</td>
<td></td>
</tr>
<tr>
<td>s2s + PE\textsubscript{abs}</td>
<td>78.15*</td>
<td></td>
</tr>
<tr>
<td>s2s + PE\textsubscript{rel}</td>
<td>78.06*</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Precision (%) on the SwDA corpus.

<table>
<thead>
<tr>
<th>Dialogue Act</th>
<th>s2s</th>
<th>s2s + PE\textsubscript{abs}</th>
<th>s2s + PE\textsubscript{rel}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statement-non-opinion</td>
<td>87.30</td>
<td>87.50 (+0.20)</td>
<td>87.69 (+0.39)</td>
</tr>
<tr>
<td>Backchannel</td>
<td>90.26</td>
<td>89.95 (-0.31)</td>
<td>90.37 (+0.12)</td>
</tr>
<tr>
<td>Statement-opinion</td>
<td>65.95</td>
<td>67.31 (+1.36)</td>
<td>66.67 (+0.71)</td>
</tr>
<tr>
<td>Uninterpretable</td>
<td>81.27</td>
<td>81.78 (+0.51)</td>
<td>81.23 (-0.03)</td>
</tr>
<tr>
<td>Agree or Accept</td>
<td>63.05</td>
<td>64.86 (+1.81)</td>
<td>63.69 (+0.64)</td>
</tr>
<tr>
<td>Appreciation</td>
<td>81.95</td>
<td>82.58 (+0.63)</td>
<td>81.95 (±0.00)</td>
</tr>
<tr>
<td>Yes-No-Question</td>
<td>81.50</td>
<td>81.83 (+0.33)</td>
<td>81.67 (+0.17)</td>
</tr>
<tr>
<td>Yes Answers</td>
<td>72.77</td>
<td>75.64 (+2.88)</td>
<td>76.27 (+3.51)</td>
</tr>
<tr>
<td>Conventional-closing</td>
<td>93.41</td>
<td>93.48 (+0.07)</td>
<td>93.19 (-0.22)</td>
</tr>
<tr>
<td>Wh-Question</td>
<td>76.25</td>
<td>74.33 (-1.93)</td>
<td>76.98 (+0.73)</td>
</tr>
</tbody>
</table>

Table 4: Precision (%) for the top 10 most frequent DA labels.

3.2 Results

Table 3 shows the results of the experiment, wherein the numbers in bold represent the best scores, and * indicates that the improvement over the baseline “s2s” is statistically significant according to the Wilcoxon signed-rank test ($p \leq 0.01$). The results show that the performance of a seq2seq-based DAR model can be significantly improved by incorporating PE\textsubscript{abs} or PE\textsubscript{rel}. In particular, PE\textsubscript{abs} and PE\textsubscript{rel} increase precision by 0.31 and 0.22, respectively. This demonstrates the effectiveness of our proposed method.

4 Analysis

4.1 Precision for Each DA Label

To analyze the effectiveness of the proposed method, we examined the precision of each model for each DA label. For simplicity, we evaluated the top 10 most frequent DA labels on the SwDA corpus. Table 5 shows the top 10 DA labels, accompanied by the number of occurrences. For the top 10 labels, Figure 2 shows the histograms of the label’s absolute and relative positions in the training data, where the vertical axis is the number of occurrences, and the horizontal axis is the relative or absolute position. The relative positions are normalized by the dialogue length.

Table 5: The number of occurrences of the top 10 most frequent DA labels.
Figure 2: Histograms of absolute and relative positions for the top 10 most frequent DA labels. The relative positions are scaled from 0 to 1 according to the dialogue length.

Table 6: Distribution of dialogue length.

4.2 Analysis of Similar DA Labels in Appearance

One of the difficulties in DAR is the recognition of DA labels in a confused group (Kumar et al., 2018; Bothe et al., 2018). In SwDA, “Backchannel,” “Agree or Accept,” and “Yes Answer” are easily confused with each other, and so are “Wh-Question,” “Open-Question,” and “Rhetorical-Question.” This is because these labels have common expressions (e.g., “Yes” and “Yeah” may belong to “Backchannel,” “Agree or Accept,” and “Yes Answer”). In this section, we analyze the performance of the proposed models for DA labels in a confused group.

Figure 3 shows the confusion matrix of “Backchannel,” “Agree or Accept,” and “Yes Answer.” As can be seen in Figure 3, the misrecognition as “Yes Answer” is widely reduced and the precision of “Yes Answer” is highly improved. This might be because “Yes Answer” has a tendency to appear at the beginning part, which is different from the appearance patterns of the other two labels (see Figure 2). This demonstrates that the proposed methods are effective to DA labels in a confused group as well.

Figure 4 shows the confusion matrix of “Wh-Question,” “Open-Question,” and “Rhetorical-Question,” and Figure 5 shows the histograms of the absolute positions for the three labels in addition to those of the relative positions in the training data. As presented in Figure 4, for PE_{rel}, the precision of “Wh-Question” is improved, but the misrecognition of “Wh-Question” as “Open-Question” increases. This might be because “Wh-Question” has a tendency of relative occurrence positions, and so does “Open-Question.” As for PE_{abs}, the
4.3 Impact of Dialogue Length

We analyze the effectiveness of the proposed method on various dialogue lengths. We divide the test data into four groups according to the dialogue length and measure precision on each group. Tables 6 and 7 show the statistics of each group and the results, respectively. In Table 7, the numbers in bold represent the best scores, and * indicates that the improvement over the baseline “s2s” is statistically significant according to the Wilcoxon signed-rank test ($\alpha \leq 0.01$).

Table 7 shows that the precision of “s2s” tends to decrease as the dialogue length increases. In contrast, both proposed models alleviate the tendency of “s2s” and preserve precision for long dialogues. Additionally, Table 7 shows that the proposed models statistically significantly outperform “s2s” on the group with the dialogue lengths of 300 or more. This indicates that our proposed models are effective for long dialogues.

In Table 7, the improvement of “s2s+PE_{rel}” over “s2s” is greater than that of “s2s+PE_{abs}.” This indicates that “s2s+PE_{rel}” could successfully encode positional information of utterances with large absolute position by normalizing their positions.

5 Conclusions

In this paper, we have proposed an utterance position-aware approach for a seq2seq-based DAR
Table 7: Precision (%) according to dialogue length.

<table>
<thead>
<tr>
<th>Model</th>
<th>len ≤ 100</th>
<th>100 &lt; len ≤ 200</th>
<th>200 &lt; len ≤ 300</th>
<th>300 &lt; len</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline Model</td>
<td>s2s</td>
<td>79.56</td>
<td>78.56</td>
<td>77.29</td>
</tr>
<tr>
<td>Proposed Model</td>
<td>s2s + PEabs</td>
<td>79.37</td>
<td>78.86</td>
<td>77.53*</td>
</tr>
<tr>
<td></td>
<td>s2s + PERel</td>
<td>79.37</td>
<td>78.68</td>
<td>77.40</td>
</tr>
</tbody>
</table>

Figure 5: Histograms of absolute and relative positions for “Wh-Question,” “Open-Question,” and “Rhetorical-Question.” The relative positions are scaled from 0 to 1 according to the dialogue length.

This work was partially supported by JSPS KAKENHI Grant Number JP21K12031.

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Number JP21K12031.

References


Tell Me What You Read: Automatic Expertise-Based Annotator Assignment for Text Annotation in Expert Domains

Hiyori Yoshikawa, 1 Tomoya Iwakura, 1 Kimi Kaneko, 2* Hiroaki Yoshida, 1 Yasutaka Kumano, 3 Kazutaka Shimada, 4 Rafal Rzepka, 5 Patrycja Swieczkowska 5

1 Fujitsu Limited, Tokyo, Japan 2 Toyota Motor Corporation, Aichi, Japan
3 G-Search Limited, Tokyo, Japan 4 Kyushu Institute of Technology, Fukuoka, Japan
5 Hokkaido University, Hokkaido, Japan

Abstract

This paper investigates the effectiveness of automatic annotator assignment for text annotation in expert domains. In the task of creating high-quality annotated corpora, expert domains often cover multiple sub-domains (e.g., organic and inorganic chemistry in the chemistry domain) either explicitly or implicitly. Therefore, it is crucial to assign annotators to documents relevant with their fine-grained domain expertise. However, most of existing methods for crowdsourcing estimate reliability of each annotator or annotated instance only after the annotation process. To address the issue, we propose a method to estimate the domain expertise of each annotator before the annotation process using information easily available from the annotators beforehand. We propose two measures to estimate the annotator expertise: an explicit measure using the predefined categories of sub-domains, and an implicit measure using distributed representations of the documents. The experimental results on chemical name annotation tasks show that the annotation accuracy improves when both explicit and implicit measures for annotator assignment are combined.

1 Introduction

Preparation of training data has been a critical issue in applying the supervised and semi-supervised methods to real world problems. Training data construction is often quite costly and the quality is hard to assure, especially when the task requires expert knowledge of the target domain such as chemistry and medicine. A possible solution to alleviate the lack of annotated data is the use of a crowdsourcing platform. Crowdsourced annotation has proven to be successful for lowering the annotation costs in tasks that require general knowledge, such as POS tagging (Hovy et al., 2014), textual entailment, word sense disambiguation (Snow et al., 2008), or recognition of general named entities like PERSON and ORGANIZATION (Finin et al., 2010; Nguyen et al., 2017). Meanwhile, there is limited success in using crowdsourcing for annotation in expert domains mainly because of the poor annotation quality made by annotators with low domain expertise. In the medical domain, for example, Nye et al. (2018) combine both non-expert crowdsourced annotators and expert annotators for text annotation.

We hypothesize that a key challenge of crowdsourced annotation in expert domains lies in the difficulty of estimating the candidate annotators’ expertise in terms of the relevance to the documents to be annotated. Some crowdsourcing platforms such as Amazon Mechanical Turk 1 provide the feature to specify workers that have the expertise in particular domains. However, it is still not optimal for annotation tasks of documents such as scientific papers, as they tend to require expertise in various specific sub-domains. For example, inorganic chemistry and drug discovery fields belong to a broader category of the chemical domain. Annotation on drug discovery papers would be difficult for experts in inorganic chemistry because it substantially differs from the drug discovery field. However, limiting the candidate annotators to the experts in drug discovery would result in insufficient number of annotators that is not enough for obtaining a large size annotated corpus within limited time. Moreover, it is even hard to identify the required domain knowledge to understand each document with such high granularity.

A practical solution to this issue would be to find a way to estimate the relevance between an annotator and a document and to assign annotators to documents based on the relevance between

---

1https://www.mturk.com/
them. It is expected that even if we can not recruit a sufficient number of annotators for each particular sub-domain, we can assign annotators with expertise in different but relevant sub-domains and obtain better quality annotations.

This motivated us to investigate the annotator assignment problem for text annotation in expert domains. The main question is whether we can improve the overall text annotation quality by using not only explicit information such as the annotators’ major fields of study but also implicit information estimated by other kinds of information available before the annotation process. To answer the question, we conduct a series of experiments on chemical name annotation tasks in academic paper abstracts from various sub-fields of chemistry. The annotators are graduate students in chemistry departments, so they have expertise in different chemistry sub-domains. As the explicit knowledge of the annotators’ expertise, we asked the annotators their fields of expertise from predefined categories. We also asked them to submit information about the literature strongly related to their study. We use the literature information to estimate more fine-grained implicit relevance between the annotated document and the annotators that can not be captured by the explicit information. The experimental results indicate that combining both the explicit and implicit information helps estimate the document-annotator relevance and results in better annotation quality.

Our approach is orthogonal to the annotation aggregation methods (Nguyen et al., 2017; Li et al., 2014) commonly used in the crowdsourced annotation. While the annotation aggregation methods improve the annotation quality by aggregating the annotations from different annotators after the annotation process, our approach improve the annotation by assigning more relevant domain experts to each text before the annotation process. The difference comes from the fact that most of the studies on crowdsourced annotation focus on annotation tasks that require general knowledge or a single expert domain, where the assignment of annotators has relatively small effect on the final annotation quality. Thus, it is possible that our approach and the annotation aggregation methods complement each other. Our approach is also different from the existing methods to select the best annotators which requires training based on the previous annotation results by the candidate annotators (Donmez et al., 2010; Kamar et al., 2013; Kamar and Horvitz, 2015; Tran-Thanh et al., 2014). We believe that estimation of the relevance between documents and annotators can also be helpful for such methods as prior knowledge.

2 Related Work

Preserving the quality of the crowdsourced work is difficult as annotators possess very diverse abilities, skills, or interests (Daniel et al., 2018). For this reason, many researchers have proposed their methods for improving the quality of annotations or maintaining it without increasing the cost. Some strategies deal with the problem of the ground truth absence – from increasing the number of annotators (Sheng et al., 2008), through intelligently weighting them based on their inferred expertise (Donmez et al., 2009; Raykar et al., 2010; Welinder et al., 2010) to recruiting a small group of top quality workers (Zhao et al., 2013; Li et al., 2014; Li and Liu, 2015; Carvalho et al., 2016) or seeing how they learn over time (Pan et al., 2016). When it comes to methods used for selecting the best annotators, Donmez et al. (2010) utilize a sequential Bayesian estimation algorithm for continuous tracking and selecting the best annotators over time. Markov Decision Process can be used to model agent conduct during the consensus tasks and to predict a candidate annotator’s work (Kamar et al., 2013; Kamar and Horvitz, 2015). Tran-Thanh et al. (2014) introduce a recruiting algorithm based on a variation of the multi-armed bandit model (MAB) outperforming previous methods by up to a remarkable 300%. Recommendation of tasks to workers has also been studied with use of a worker’s task browsing history (Yuen et al., 2015), or taking into account implicit negative feedback (Lin et al., 2014).

However, all these approaches focus on annotators, not the target data. Unlike these methods, our approach is to assign documents to experts in potentially different sub-domains so that their expert domains are as close as possible to those of the assigned documents. An example of annotation improvement in a scenario requiring specialized expertise (clinical NLP) is given in Dumitranche et al. (2015), where authors pay attention to documents being annotated. The main problem they aim to solve is not only the lack of ground truth for training and benchmarking but also ambiguity in the target documents. They have shown that, with proper processing, the crowd performs just
as well as medical experts in terms of the quality and efficacy of annotations, while being cheaper and more readily available. However, their results indicate that at least ten workers per sentence are needed to get the highest quality annotations, while we aim at acquiring high-quality results with fewer workers by assuring assignments of targets well-fit to the annotators.

Assignment of experts has been studied in the context of paper-reviewer assignment problem at academic conferences (Dumais and Nielsen, 1992; Charlin and Zeme, 2013; Dumais and Nielsen, 2016). These methods use previous publications of reviewers and those publications are used to build reviewers’ profiles. Methods for building profiles include not only use of words in publications, but also Latent Semantic Indexing (Salton and McGill, 1983) and Latent Dirichlet Allocation (Blei et al., 2003). Submitted papers are also encoded with the same methods for building reviewer profiles. Then, the similarity between submitted papers and each reviewer’s profile is calculated based on the encoded representations and assignment is done just with the calculated similarity or by casting assignment as an Integer Linear Programming problem (Karimzadehgan and Zhai, 2009). Our work is in the similar spirit as this line of work. The experimental results indicate that such an approach is also promising in selecting annotators for expert domain annotation tasks.

To sum up the related work: current crowdsourcing-based annotation methods integrate several annotation results into one final annotation, utilize machine learning with several annotation results, or target experts where the most skillful annotators are automatically discovered. To the best of the authors’ knowledge, the proposed task of automatic assignment of documents for annotation is the first of its kind. Besides, this is the first research that evaluates the effectiveness of consideration of annotator’s expertise in quality.

3 Expertise-Based Annotator Assignment

3.1 Problem Description

We consider the annotator assignment problem for text annotation, where the goal is to assign the most relevant annotators to the given document for better annotation quality. In particular, we consider annotation tasks on documents that require special knowledge for full understanding such as scientific papers and clinical records. We also assume that the required sub-domain of knowledge differs depending on the document to be annotated. For the chemistry domain, the sub-domains may include inorganic chemistry, drug discovery, and so on. As clues of annotator’s expertise, we consider two types of information: explicit expertise such as predefined sub-domains and implicit expertise that is estimated from the information available before the annotation process and potentially represents more fine-grained relevance between documents and annotators. The implicit expertise can be helpful not only for complementing explicit expertise, which might not be available or might be insufficient but also for capturing the similarity between different sub-domains to estimate the relevance of the papers to the annotators in different sub-domains. We describe these two types of expertise in more details in the following section.

In this paper, we evaluate our approach on the chemical name annotation task using chemistry paper abstracts. In what follows, we describe our proposed method based on this specific task. However, the method can be generalized to different domains and tasks where similar type of prior information about the target documents and the candidate annotators is available. It is worth noting that our approach depends only on the type of information about the documents and the annotators available before the annotation process, and is independent of the type of the annotation task.

3.2 Prior Information of Annotator Expertise

3.2.1 Explicit Expertise

For explicit expertise, we assume existence of predefined categorical labels representing expert sub-domains. Both the documents to be annotated and the candidate annotators should be associated with categorical labels representing the sub-domains of the documents’ or the annotators’ expertise. In practice, the labels can be the conference venues of papers, IPC classification of patents, and so on. Multiple labels can be associated with each document and annotator. We use the binary indicator $I_{cat}(i, j) \in \{0, 1\}$ for the explicit expertise-based relevance score between the annotator $i$ and the document $j$, which indicates whether the annotator $i$’s areas of expertise include at least one of the categories of the document $j$ or not.
3.2.2 Implicit Expertise

In addition to the explicit expertise measure, we propose document-annotator relevance based on implicit expertise that captures more fine-grained and multi-dimensional aspects of the domain expertise. We assume that we have access to a small subset of documents that each annotator is the most familiar with from the same domain as those to be annotated (hereinafter called annotators’ documents). This requires the candidate annotators to present additional information, but it is not hard for annotators to present. For example, it can be a few papers relevant to their fields of study.

We encode both the annotators’ documents and the documents to be annotated into a low-dimensional vector in some way. These document representations include semantic knowledge about the specific sub-domains of the documents and the annotations that can not be captured by the categorical information alone. Suppose that we have a set of candidate annotators \( I \) and a set of documents to be annotated \( J \). Let \( d^{(i)}_{il}(t) \) \((t \in \{1, \ldots, L\})\) be the representation of the \( i \)-th annotator’s documents (assuming that each annotator presents \( L \) relevant documents), and \( d^{(j)}_{j}(t) \) be the representation of the \( j \)-th document to be annotated. For annotators that have expertise in multiple sub-areas, each of the annotators’ documents may represent different aspects of their expertise. Therefore, we compare two different ways to aggregate these paper representations and compute the final relevance score. The first one is to compare each document to be annotated with the average of the document representations of the annotator’s documents:

\[
s_{av}(i, j) = \text{sim}(\frac{1}{L} \sum_{l=1}^{L} d^{(s)}_{il}, d^{(t)}_{j}),
\]

where \( \text{sim}(\cdot, \cdot) \) denotes the cosine similarity. The second one is to compute the similarity between each of the annotator’s documents and the document to be annotated, and take the score from the most relevant one:

\[
s_{\text{nearest}}(i, j) = \max_{l} \text{sim}(d^{(s)}_{il}, d^{(t)}_{j}).
\]

3.2.3 Relevance Scores

In our experiments, we evaluate the effect of combining these explicit and implicit relevance measures. We employ the following combinations:

- \( \text{cat}: I_{\text{cat}}(i, j) \),
- \( \text{catsim-av}: I_{\text{cat}}(i, j) + s_{av}(i, j) \), and
- \( \text{catsim-nearest}: I_{\text{cat}}(i, j) + s_{\text{nearest}}(i, j) \).

Note that as \( I_{\text{cat}}(i, j) \) takes the value 0 or 1 and \( s_{av}(i, j) \) and \( s_{\text{nearest}}(i, j) \) is between \([0, 1]\), the agreement based on the explicit expertise has priority over that of implicit expertise in terms of the final relevance score. Therefore, the implicit expertise-based relevance is expected to work as the auxiliary information that provides the knowledge that can not be captured by the explicit expertise-based measure.

3.3 Calculating Annotator Assignment

The document-annotator assignment is calculated by solving the following Integer Linear Programming (ILP) problem:

Maximize \[ \sum_{i \in I, j \in J} s_{i,j} \cdot x_{i,j} \] (3)

\[ \text{s.t. } k_{\text{min}} \leq \sum_{j \in J} x_{i,j} \leq k_{\text{max}} \quad \forall \ i \in I \] (4)

\[ \sum_{i \in I} x_{i,j} = n_{\text{ann}} \quad \forall j \in J, \] (5)

\[ x_{i,j} \in \{0, 1\} \quad \forall i \in I, \forall j \in J, \] (6)

where \( s_{i,j} \) is one of the relevance scores between the \( i \)-th annotator and the \( j \)-th document introduced above. The problem is to find the assignment \( x_{i,j} \) that maximizes the total relevance scores of the assigned pairs of documents and annotators. \( x_{i,j} \) is the indicator variable that becomes 1 if the \( i \)-th annotator is assigned to the \( j \)-th document, and 0 otherwise. The first constraint represents the maximum \( k_{\text{max}} \) and the minimum \( k_{\text{min}} \) numbers of documents that can be assigned to a single annotator. In order to evenly assign documents to annotators, we use \( k_{\text{min}} = \lceil |J|/|I| \rceil \) and \( k_{\text{max}} = k_{\text{min}} + 1 \) in our experiment. The second constraint means that each document should be assigned to \( n_{\text{ann}} \) annotators. \( n_{\text{ann}} \) is set to 1 in our experiment.

4 Experimental Settings

We evaluated the proposed annotator assignment method on a chemical name annotation task using the abstracts of chemistry papers.

4.1 Datasets and Guidelines

We used chemistry papers in two different languages: English and Japanese. The papers are selected randomly from the predefined categories to construct a dataset of approximately 500 abstracts for each language. The sub-domain categorization used for each language is shown in Table 1.
4.1.1 English Papers

We used CHEMDNER corpus (Krallinger et al., 2015) for the English annotation task. It consists of 10,000 abstracts of chemistry-related publications that were chosen according to journal titles. These titles helped us assign categories to English texts (listed in Table 1; multiple categories for each paper were allowed). Each abstract in the CHEMDNER corpus was annotated with seven types of chemical term classes defined by the CHEMDNER task organizer. To evaluate our annotation, we used paper abstracts from the CHEMDNER test set which consists of 3,000 abstracts. We randomly sampled 504 abstracts while maintaining an approximately even number of papers in every category.

For annotation evaluation, we followed the annotation guidelines of the CHEMDNER corpus\(^2\). The labels in the CHEMDNER test set were used as the gold standard. The original paper reports an inter-annotator agreement of 85.26%.

4.1.2 Japanese Papers

We obtained Japanese chemistry paper abstracts from JDREAM III\(^3\) repository of scientific publications based on their categories as of September, 2017. JDREAM III provides predefined categories corresponding to the first eight of those in Table 1. We retrieved 2,500 papers for each category, and the total number of papers was 20,000. We applied to them an in-house chemical named entity recognizer to select the paper abstracts that were likely to include some chemical compound names or chemistry-related terms. Finally, we randomly sampled 520 abstracts from the selected abstracts so that the number of abstracts in each category was 65.

For annotation of Japanese texts, we prepared a new annotation guideline defining 20 mention types for chemical entities. We defined twelve entity types for chemical substances including organic and inorganic molecules and eight entity types for chemistry-related concepts such as drug names, products, properties, and numerical expressions. An example of annotated document is shown in Appendix A.

To prepare the gold standard data, we employed 17 in-house expert annotators to annotate chemistry-related terms in these paper abstracts. In order to create a reliable gold standard, each abstract was checked by three annotators. First, two annotators annotated each abstract, and then the remaining annotator, who had not annotated the abstract, checked and integrated the annotations. The inter-annotator agreement between the two annotators was 0.449 in terms of exact match entity-level F\(_1\) score.

4.2 Annotators

We employed 49 graduate students who major in chemistry at their universities. 40 and 20 students performed annotation for English and Japanese, respectively. Some students annotated both English and Japanese texts.

As prior information for estimating the subdomain expertise of each student, we asked the students to provide the following information:

- Expertise category as listed in Table 1. Multiple choice was allowed.
- Titles and abstracts of five scientific papers each that were relevant to their study.

\(^2\)To be precise, we used the annotation guidelines for CHEMDNER patent corpus as CHEMDNER server was not accessible during our experiment. However, the guidelines of the patent corpus defined the same seven mention types and in our preliminary evaluation we confirmed that the differences between them are small.

\(^3\)https://jdream3.com/
The distribution of their expertise categories is shown in Table 1.

All annotations were conducted using the BRAT annotation tool (Stenetorp et al., 2012). For both English and Japanese papers, we provided the annotators with the same annotation guidelines that had been used to construct the gold-standard dataset (i.e. the CHEMDNER patent guideline for English and our new guideline for Japanese). In order to alleviate the problem of lower quality caused by inexperience in annotation and to assure proper understanding of the annotation guidelines, we first trained all the annotators by asking them to annotate the same five paper abstracts in a trial stage. After that, we asked each annotator to annotate the assigned abstracts.

In addition to cat, catsim-av and catsim-nearest, we also evaluated the performance of non-experts who have no expertise in chemistry. The non-experts’ fields of expertise were other than chemistry, such as architectonics, physics and electricity. Abstracts assignment to the non-experts was random.

4.3 Annotator Assignment

As the measure of the implicit expertise, we calculated semantic representations of submitted paper abstracts using a simple sentence embedding method. For each paper, we averaged the sum of embeddings of all content words in the title and abstract of the paper after normalizing the Euclidean norm of each word vector to 1. We used 200-dimensional word embeddings trained with word2vec (Mikolov et al., 2013) for both English and Japanese. Training data for the English word embeddings was the whole CHEMDNER corpus in addition to the titles and abstracts of papers obtained from MEDLINE 2017 version that appear in the same set of journals used in the CHEMDNER corpus. Training data for the Japanese word embeddings were the titles and abstracts of 20,000 papers from JDREAM III.

The ILP problem for the assignment was solved with the COIN CBC solver of PuLP and the optimum solutions were obtained for all the assignments. The optimization result was obtained within a few seconds. Each annotator was asked to annotate the combined set of abstracts assigned by the three assignment methods. The order of the abstracts given to each annotator was random and the annotators were not informed which method was used to assign each abstract. As some of the annotation jobs were canceled after the assignment, we ended up doing the evaluation with 323 paper abstracts for English and 375 paper abstracts for Japanese for which we obtained results from all the assignment methods.

5 Results

5.1 Main Results

Table 2 shows the annotation performance with different assignment methods. We use the standard evaluation metrics for named entity tagging, i.e. recall, precision and F-measure \( F_1 \) based on the exact match of the tagged entities compared with the gold standard.

The performance by non-experts is significantly worse than those of domain experts, indicating the importance of domain expertise for our evaluation task. On the other hand, the performance of expertise-based assignments is higher than the agreement between experts (0.449) for the Japanese documents. \(^7\) Compared to (cat), which only uses the explicit expertise for assignment, catsim-nearest showed higher \( F_1 \) for both English and Japanese data set. catsim-av demonstrated higher \( F_1 \) for English but lower \( F_1 \) for annotations in Japanese. A possible reason why catsim-nearest measure generally performs better than catsim-av is that the former is better at capturing expertise over multiple sub-domains. When an annotator has expertise in multiple sub-domains, catsim-av represents their expertise with a single vector by averaging the representations of all the annotator’s documents. On the other hand, catsim-nearest keeps the individual representations of all the annotator’s documents and uses the one that is the most relevant to each target document for the relevance score. The improvement is relatively higher in recall than in precision for both English and Japanese, indicating that sub-domain expertise helps annotators in detecting more technical terms in the documents.

---

\(^4\)We used NLTK (Bird, 2006) and Mecab (Kudo et al., 2004) tokenizers for English and Japanese, respectively, to identify content word using part-of-speech tags.


\(^6\)https://github.com/coin-or/pulp

\(^7\)Although the reported inter-annotator agreement of the English corpus is 85.26%, it is hard to directly compare the result with ours as no details are provided for the calculation of the agreement. In addition, the annotators of the corpus were also involved in revising the annotation guideline, indicating higher proficiency in the guideline.
Table 2: The annotation quality for each assignment measured with recall (R), precision (P) and F1 scores.

<table>
<thead>
<tr>
<th></th>
<th>English</th>
<th>Japanese</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>P</td>
</tr>
<tr>
<td>non-expert</td>
<td>0.313</td>
<td>0.495</td>
</tr>
<tr>
<td>cat</td>
<td>0.525</td>
<td>0.481</td>
</tr>
<tr>
<td>catsim-av</td>
<td>0.550</td>
<td>0.496</td>
</tr>
<tr>
<td>catsim-nearest</td>
<td>0.569</td>
<td>0.507</td>
</tr>
</tbody>
</table>

To compare the results of annotations, we employed a McNemar paired test on the labeling disagreements following the procedure introduced in Sha and Pereira (2003). As in Kudo et al. (2004) for morphological analysis, we compared the results based on character-based IOB2 format (Tjong Kim Sang and Veenstra, 1999) instead of the usual word-based version, because the endings or beginnings of chemical terms do not always correspond to word boundaries. The results confirmed statistical significance ($p < 0.01$) of the differences between cat and catsim-nearest for both English and Japanese.

5.2 Error Analysis

Table 3 shows the statistics of the annotation results with respect to the types of errors. We classified the annotation errors into four types: incorrect tags assigned to characters other than chemical terms (Ex), chemical terms which were not annotated (Miss), annotations whose spans are different from the correct annotation (SD), and annotations where spans are correct, but the type of the chemical terms is incorrect (TD). An example of each error type is shown on the right-hand side of the table. The results show that assigning more relevant domain experts to the task helps to reduce the number of missed annotations while slightly increasing the number of excessive annotations. It is understandable though that annotators are likely to give more labels to documents relevant to their expertise.

We also evaluated proposed methods by mention-wise comparison of annotation results. The results show that the annotations corresponding to implicit expertise-based assignments are better than explicit expertise-based ones in terms of the number of correctly annotated mentions. McNemar paired test showed statistical significance ($p < 0.05$) of the improvement in English results, while no significance was observed in Japanese results. We also computed the correlation between the relevance scores and the annotation accuracy as shown in Table 5. Similarly to the mention-wise comparison, we observed significant correlation between implicit expertise-based relevance scores and the corresponding F1 scores for the English task, while no significant correlation was observed for the Japanese task. Possible reasons why the proposed method is less efficient in the Japanese task is discussed in the following section.

6 Discussion and Conclusion

Good quality corpora of specialized scientific texts could become important not only for specialists from various fields, but also for the promising AI subfield of automatic scientific discovery. This paper proposes a novel method for automatic annotation task assignment based on the expertise of the annotator estimated with scientific paper abstracts that the annotator has presented as relevant to their own research interests. The experimental annotation results of English and Japanese chemistry-related paper abstracts showed that our method contributes to higher accuracy compared to the annotation by non-experts and use of predefined technical field categories.

We could not cover all possible experimental settings (e.g. assignment using only implicit expertise) due to the time and budget restrictions. In our experimental design it is required that all assignments to be compared are calculated on the same set of annotators and then annotated together, which makes adding other experimental settings afterwards difficult. Improvement in experimental design is needed in order to compare different factors more flexibly.

An interesting observation is that the cause of annotation errors is not always the lack of annotator expertise. For example, we found that errors which result from poor understanding of annotation guidelines are not negligible. For the Japanese task,
### Table 3: (Left) The number of correct annotations (Corr), incorrect tags assigned to characters other than chemical terms (Ex), chemical terms which were not annotated (Miss), annotations whose spans are different from the correct annotation (SD), and annotations where spans are correct, but the type of the chemical terms is incorrect (TD). (Right) Examples of errors for each error type.

<table>
<thead>
<tr>
<th></th>
<th>Corr</th>
<th>Ex</th>
<th>Miss</th>
<th>SD</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>non-expert</td>
<td>860</td>
<td>336</td>
<td>1316</td>
<td>280</td>
<td>306</td>
</tr>
<tr>
<td>cat</td>
<td>1441</td>
<td>787</td>
<td>519</td>
<td>395</td>
<td>430</td>
</tr>
<tr>
<td>catsim-av</td>
<td>1511</td>
<td>798</td>
<td>476</td>
<td>325</td>
<td>459</td>
</tr>
<tr>
<td>catsim-nearest</td>
<td>1563</td>
<td>806</td>
<td>467</td>
<td>409</td>
<td>357</td>
</tr>
<tr>
<td>Japanese</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>non-expert</td>
<td>3484</td>
<td>1940</td>
<td>1652</td>
<td>2080</td>
<td>1174</td>
</tr>
<tr>
<td>cat</td>
<td>3655</td>
<td>1165</td>
<td>1912</td>
<td>1717</td>
<td>961</td>
</tr>
<tr>
<td>catsim-av</td>
<td>3682</td>
<td>1229</td>
<td>1818</td>
<td>1746</td>
<td>1016</td>
</tr>
<tr>
<td>catsim-nearest</td>
<td>3734</td>
<td>1211</td>
<td>1789</td>
<td>1649</td>
<td>1062</td>
</tr>
</tbody>
</table>

### Table 4: Correlation matrices comparing two of three annotation results by mentions. The upper-left cell (T–T) of each result indicates the number of mentions which were correctly annotated by both methods, the upper-right cell (T–F) corresponds to the mentions which were correctly annotated by cat but were not by the other method, and so on.

<table>
<thead>
<tr>
<th></th>
<th>catsim-av</th>
<th>catsim-nearest</th>
</tr>
</thead>
<tbody>
<tr>
<td>T–T</td>
<td>T–F</td>
<td>F–F</td>
</tr>
<tr>
<td>cat</td>
<td>T 1029 412</td>
<td>T 1060 381</td>
</tr>
<tr>
<td></td>
<td>F 482 824</td>
<td>F 503 803</td>
</tr>
<tr>
<td>catsim-av</td>
<td>T 2777 878</td>
<td>T 2736 919</td>
</tr>
<tr>
<td>catsim-nearest</td>
<td>F 905 3474</td>
<td>F 998 3381</td>
</tr>
</tbody>
</table>

### Table 5: Spearman rank correlation coefficients between task assignment scores and F1 scores on the corresponding annotation results. (En) and (Ja) correspond to experiments on English and Japanese abstracts, respectively. * means that the relationship is statistically significant at \( p < 0.05 \).

<table>
<thead>
<tr>
<th></th>
<th>Corr. (En)</th>
<th>Corr. (Ja)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>0.02</td>
<td>0.00</td>
</tr>
<tr>
<td>catsim-av</td>
<td>0.12*</td>
<td>0.09</td>
</tr>
<tr>
<td>catsim-nearest</td>
<td>0.14*</td>
<td>-0.01</td>
</tr>
</tbody>
</table>

In which we failed to find any significant improvement, as many as 20 of mention types might have made the annotation task too complicated for annotators that are domain experts but not experts in linguistic annotation. As shown in Table 3, a large number of annotation errors in the Japanese task are span difference (SD). This type of errors is often less relevant to domain expertise: an example is “Sb(CN)3(2,2’-bipy)” vs. “[Sb(CN)3(2,2’-bipy)]”.

Another future work topic is designing an annotation framework for domains requiring expertise knowledge. For experiments described in this paper, we recruited students who major in chemistry with the help of university faculties. However, in order to continue building corpora for domains requiring expertise knowledge, this procedure is not optimal. In fact, it took about three weeks just to hire annotators for the experiments. In the future, it is necessary to develop not only accurate assignment methods, but also an annotation framework for expert domains. In order to apply our method to documents other than scientific papers, it is also necessary to find alternatives for the “relevant scientific papers” that represent the annotators’ domain expertise. The application fields include Q&A texts and blogs that feature domain specific topics. It is reported that annotation accuracy on such texts gets worse when the task requires specific knowledge of cartoons and TV programs and so on (Komiya et al., 2016). For such cases, it might be helpful to use cartoon titles, TV-show titles, news articles, blog posts, etc., recently read or watched by annotators.

### Acknowledgments

A part of this work was carried out in the RIKEN Center for Advanced Intelligence Project.

### References

Steven Bird. 2006. **NLTK: The Natural Language Toolkit.** In *Proceedings of the COLING/ACL 2006*


Pinar Donmez, Jaime Carbonell, and Jeff Schneider. 2010. A probabilistic framework to learn from multiple annotators with time-varying accuracy. In Proceedings of the 2010 SIAM International Conference on Data Mining, pages 826–837. SIAM.


Zhou Zhao, Da Yan, Wilfred Ng, and Shi Gao. 2013. A transfer learning based framework of crowd-selection on twitter. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1514–1517. ACM.

A Annotation of Japanese Documents

Figure 1 shows an example of an annotated part of a Japanese paper.

Figure 1: An example of annotated Japanese text.
Abstract

Neural sequence-to-sequence (Seq2Seq) models and BERT have achieved substantial improvements in abstractive document summarization without and with pre-training, respectively. However, they sometimes repeatedly attend to unimportant source phrases while mistakenly ignore important ones. We present new reconstruction mechanisms on two levels to alleviate this issue. The sequence-level reconstructor reconstructs the whole source document from the hidden layer of the target summary, while the word embedding-level one rebuilds the average of word embeddings of the source at the target side to guarantee that as much critical information is included in the summary as possible. Based on the assumption that inverse document frequency (IDF) measures how important a word is, we further leverage the IDF weights in our embedding-level reconstructor. The proposed frameworks lead to promising improvements for ROUGE metrics and human rating on both the CNN/Daily Mail and Newsroom summarization datasets.

1 Introduction

Single document summarization is designed to automatically compress a document into its short version without changing the main idea. The summarization task is generally divided into two categories: extractive methods that copy certain sentences or phrases directly from the source text, and abstractive methods that paraphrase the source text by using novel words. Abstractive summarization has the potential to produce summaries in the same way that humans do.

Recent years have witnessed significant progress in the abstractive summarization task performed by Seq2Seq models, which encode a source text and decode its summary. A hybrid of the extractive and abstractive techniques, called pointer-generator network (PGN) (Gu et al., 2016; See et al., 2017), has been widely used as a basis for many studies (Gehrmann et al., 2018; Shen et al., 2019; Shi et al., 2019) thanks to its capability of copying words from the source document and generating new words. With the rich contextual representations, pre-trained encoders (Devlin et al., 2018) have also improved the state-of-the-art on this task (Liu and Lapata, 2019; Raffel et al., 2020). However, the conventional PGN and BERT face two main problems: 1) When generating summaries, PGN tends to frequently pay attention to some source parts while neglecting other parts, which are regarded as over- and under-attention respectively. Therefore, the model requires a mechanism to make sure that as much salient information is transformed from the source to the target as possible. 2) Additionally, BERT is pre-trained for sentences (Xu et al., 2020), thus it is deficient in distinguishing key points from non-key points within long-range dependencies, which leads to inconsequential phrases or words being copied into the summary. This is mainly due to the lack of ability to recognize topic-signifying words from the document (Baziotis et al., 2019).

Similar problems have been encountered in neural machine translation (NMT) (Tu et al., 2016). Tu et al. (2017) developed an encoder-decoder-reconstructor framework with a two-step process that first translates a sentence into another language and then reconstructs the translation back to the source language. The newly added reconstructor rewards the model when it correctly reconstructs the input source sentence from the decoder hidden layer, which forces the salient information to be transferred from the source to the target. Baziotis et al. (2019) applied this concept to unsupervised sentence compression, where the model consists of two encoder-decoder pairs so that no large text-summary dataset is needed.

Inspired by the above, we propose three recon-
struction methods for neural document summarization. The intuitive approach is a reconstructor that rebuilds the source document from the decoder hidden layer in an encoder-decoder architecture that assigns the corresponding likelihood as a reconstruction loss. When the reconstructed text is significantly different from the original one due to the model incorrectly omitting or repeating some parts to generate the summary, the reconstructor penalizes it during training. We refer to this approach as sequence-level reconstruction and choose it as our first implementation of the idea.

In contrast to the sequence-level reconstruction, reconstructing the word embeddings of source articles would be a rather simple and effective method for document summarization. The embedding-level reconstruction alleviates the length discrepancy between articles and summaries by calculating the distance between the average word embeddings of the source and target sides. We therefore propose the average word embedding reconstructor as our second reconstruction method.

We also present the third reconstructor that focuses on the saliency of words. Generally, certain words that appear frequently in documents have little importance and should not be kept in the summary. In contrast, words that signify the main idea are supposed to be maintained in the summary. As inverse document frequency (IDF) can measure how important a token is and extract salient information (Salton et al., 1983), we reconstruct the IDF-weighted word embeddings of the source at the target to keep topic-signifying words in the summary. Since the summary has far fewer tokens than the document, it is not appropriate to directly incorporate the term frequency (TF) value into our reconstructor. We therefore consider the IDF as the weight of embeddings, rather than the TF-IDF.

In this work, we mainly adopt the PGN as baseline and incorporate the above three reconstructors upon it. We then assign a loss for each reconstructor and leverage each of them as a complement to the baseline objective. On one hand, the reconstruction objectives facilitate the model to generally focus on the entire text rather than parts of it, thereby avoiding under-attention. On the other hand, the IDF-weighted method serves as a selector by giving more weights to topic-signifying words such that it can identify essential parts (Table 1) and prevent over-attention to less important words.

We performed experiments on two datasets and compared our methods with the baselines. Experimental results on the Newsroom dataset demonstrate that we outperformed the baselines by more than 2 points in ROUGE-1, 2, and L. Our methods also led to significant improvements on the CNN/Daily Mail dataset.

2 Preliminaries

In the PGN, a document $x$ that consists of a sequence of tokens $x = \{x_1, x_2, ..., x_T\}$ is fed into a bidirectional LSTM encoder, producing a sequence of hidden states $h_i$. Then a unidirectional LSTM decoder generates its corresponding summary $y = \{y_1, y_2, ..., y_T\}$ word by word with the limitation of $T \ll I$. $I$ and $T$ indicate the lengths of the source article and the summary, respectively.

The PGN adopts the attention mechanism to learn the alignment and yield target tokens simultaneously. Conditioned on decoder hidden state $s_t$ and context vector $c_t$ for the $t$-th decoding step, vocabulary distribution $P_{\text{vocab}}$ is as follows:

$$P_{\text{vocab}}^{t} = \text{softmax}(W_s s_t + W_c c_t + b_s + b_c),$$  

$$c_t = \sum_{i=1}^{I} \alpha_{t,i} h_i,  \tag{1}$$

$$\alpha_{t,i} = \exp(\alpha'_{t,i}) / \sum_{k=1}^{I} \exp(\alpha'_{t,k}),  \tag{2}$$

$$\alpha'_{t,i} = W_\alpha \tanh(W_h h_i + W_s s_t + b_\alpha),  \tag{3}$$

where $W_s$, $W_c$, $W_\alpha$, $W_h$, $W_s'$, $b_s$, $b_c$, $b_\alpha$ are trainable parameters, and $\alpha_t = \{\alpha_{t,1}, ..., \alpha_{t,I}\}$ is the
attention distribution over the source hidden states.

The PGN additionally employs the copy mechanism to decide whether to copy a word from the source document or to generate a new word through soft switch $p_{gen,t} \in [0, 1]$. $p_{gen,t}$ can be obtained by a feed-forward network whose inputs are the context vector and the encoder and decoder hidden states. The copy distribution is sampled from attention distribution $\alpha_t$, that is, the copy probability of a source token $w$ is calculated as the sum of attentions towards all occurrences of $w$. Thus, the joint distribution is calculated as

$$P(y_t) = p_{gen,t} \times P_{vocab,t}(y_t) + (1 - p_{gen,t}) \times \sum_{i:x_i = y_t} \alpha_{t,i}.$$  

During training, we use the negative log-likelihood as the loss function:

$$L_{PGN} = -\sum_{t=1}^{T} \log P(y_t).$$  

### 3 Proposed Model

We next describe the details of our proposed methods, which can be split into two main components:

- **Pointer-generator**: a neural Seq2Seq framework with the attention and copy mechanisms, as introduced in Sec. 2.

- **Reconstructor**: a module that manages to reconstruct the salient information of the original document in its summary. We put forward three independent reconstructors that will be explained in the next subsections. Moreover, our proposed approaches are applicable to any attention-based Seq2Seq summarization architectures.

#### 3.1 Sequence-level Reconstruction

The first reconstructor, as shown in Fig. 1, is expected to recover the full input sequence from the decoded summary, i.e., to reconstruct the one-hot representations of the tokens in the source document to reward the summary with the complete source information. Specifically, the reconstructor generates a reconstructed sequence $\hat{x} = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_I\}$ word by word from decoded summary sequence $y$ and decoder hidden state $s_t$. We obtain a probability distribution over the vocabulary through reconstructor hidden state $\hat{h}_t$ and inverse context vector $\hat{c}_i$:

$$\hat{P}_{vocab,i} = \text{softmax}(\hat{W}_h \hat{h}_t + \hat{W}_c \hat{c}_i + \hat{b}_s),$$  

$$\hat{c}_i = \sum_{t=1}^{T} \hat{\alpha}_{t,i} \hat{s}_t,$$  

where inverse attention $\hat{\alpha}_t$ for reconstructing step $i$ has the same structure as the original attention, except taking the decoder and reconstructor hidden states as inputs and owning independent weighted vectors. Then, we try to minimize the reconstruction loss, which is the negative log-likelihood assigned by $\hat{x}$ to the original document $x$:

$$L_{\text{recon}} = -\sum_{i=1}^{I} \log \hat{P}_{vocab,i}(\hat{x}_i = x_i).$$  

In general, the reconstruction phase can be treated as an inverse process of a standard encoder-decoder.

It is obviously impossible to reduce the loss to a low level because a summary must contain fewer tokens and less information than its original text. However, we can reasonably expect the reconstruction loss to urge the encoder-decoder to embed complete information of the source document.

#### 3.2 Word Embedding-level Reconstruction

In order to ensure the generated summary maintains a similar sequence representation with the source article, we compute the average word embeddings of $x$ and $y$, and attempt to minimize their cosine distance. For a source word $x_i$ in the input document, we simply utilize vector $e_{x_i}$ obtained from the encoder embedding layer. To represent token $y_i$, we first concatenate context vector $c_{t-1}$ and embedding $e_{y_{t-1}}$ of the word generated at the previous step. $e_{y_{t-1}}$ keeps and shares part of the word embedding information at the input side, while $c_{t-1}$ simultaneously adds new information about context at the output side. Then, a linear transformation is applied to combine above two vectors:

$$e_{y_{t}} = W_e [c_{t-1}; e_{y_{t-1}}] + b_e,$$

where $W_e$ and $b_e$ are trainable parameters. Considering the difference of the embedding representations between the document and its summary, we take the following actions: 1) the embedding matrix is shared between the encoder and
decoder for unity, 2) the method pays fair attention (weight) to each word in the sequence to restrain under-attention, and 3) to prevent the effect of length difference, we calculate the average of the word embeddings for each sequence, i.e., divide the sum by their respective lengths.

We calculate the embedding representations for original article $x$ and summary $y$ as follows:

$$r(x) = \frac{1}{I} \sum_{i=1}^{I} e_{x_i}, r(y) = \frac{1}{T} \sum_{t=1}^{T} e_{y_t}. \quad (11)$$

The reconstruction loss is then examined by the cosine similarity between summary representation $r(y)$ and source representation $r(x)$ accordingly as

$$\mathcal{L}_{\text{recon}} = 1 - \cos(r(x), r(y)). \quad (12)$$

### 3.3 IDF-weighted Embedding-level Reconstruction

The abstractive summarization task is intended to remove duplicate or unimportant words and to paraphrase the rest. The second approach, introduced above, takes the average of word embeddings (i.e., fair weights) as the goal of reconstruction. However, attending to all words equally does not suit the objective of this task. Thus, we propose an advanced version of the word embedding reconstruction by incorporating IDF's, as shown in Fig. 2.

Intuitively, some words, e.g., "the", appear in many documents, while others, e.g., "Harry Potter", are not so frequent. Therefore, words with lower IDF values usually have no specific meaning and can be omitted without confusing the main idea. Conversely, higher valued words might signify the topic of an article. This assumption allows the model to distinguish key points from non-key points, thereby avoid over-attention to less important parts. Consequently, IDF-weighted embeddings are used as an alternative, changing Eq. 11 to

$$r(x) = \frac{\sum_{i=1}^{I} \text{IDF}_{x_i} e_{x_i}}{\sum_{i=1}^{I} \text{IDF}_{x_i}}, r(y) = \frac{\sum_{t=1}^{T} \text{IDF}_{y_t} e_{y_t}}{\sum_{n=1}^{T} \text{IDF}_{y_n}}. \quad (13)$$

The same form of reconstruction loss as Eq. 12 is implemented here as well.

The IDF values can be computed on the basis of the training dataset. They are calculated separately from a corpus of original articles and reference summaries to create source and target-side dictionaries, respectively. Given a token $w$, its inverse document frequency can be obtained by

$$\text{IDF}_w = \log \frac{1 + n_d}{1 + DF(d, w)} + 1, \quad (14)$$

where $n_d$ denotes the total number of documents in the corpus and $DF(d, w)$ is the number of documents where $w$ appears. 1) At the source side, the model takes the encoder input $x_i$ as a key to search for $\text{IDF}_{x_i}$ from the source-side dictionary. 2) Whereas at the target side, the decoder outputs a probability distribution at each time step according to Eq. 5. We choose the word with the highest probability as the output and look for its corresponding $\text{IDF}_{y_t}$ in the target-side dictionary.

### 3.4 Training Loss

We use $\lambda$ as a hyperparameter to balance $\mathcal{L}_{\text{PGN}}$ with $\mathcal{L}_{\text{recon}}$. The overall loss can be defined as

$$\mathcal{L} = \mathcal{L}_{\text{PGN}} + \lambda \mathcal{L}_{\text{recon}}. \quad (15)$$

![Figure 2: IDF-weighted word embedding reconstruction model.](image)
4 Experiments

4.1 Datasets and Settings

Datasets  We carried out the experiments on two benchmark datasets, namely CNN/Daily Mail (Nallapati et al., 2016) and Newsroom (Grusky et al., 2018). Using these two datasets is challenging since we need to compress long news articles into short multi-sentence summaries. To split the CNN/Daily Mail dataset into training/validation/test sets, we followed See et al. (2017) to use the non-anonymized version. We replicated the pre-processing steps released by Shi et al. (2019) to generate the splits of the Newsroom dataset. The basic statistics of the datasets, including the splitting details, are summarized in Table 2. In both datasets, the document and summary were truncated to 400 and 100 tokens, respectively. Additionally, 50K of the most frequently occurring tokens in the training dataset were selected to form a vocabulary for both the source and target.

Evaluation metrics  We evaluated our models with the ROUGE metrics (Lin, 2004), which compare model-generated summaries with reference summaries by referring to the overlap of unigram (ROUGE-1), bigram (ROUGE-2), and longest common subsequence (ROUGE-L).

Experimental settings  We trained our models on a single GeForce RTX 2080Ti GPU (11GB RAM). 128-dimensional word embeddings with random initialization were fine-tuned during training. We utilized a single-layer bidirectional LSTM for the encoder and a unidirectional LSTM for the decoder. Both the encoder and decoder have 256-dimensional hidden states. As for the optimizer, Adam (Kingma and Ba, 2015) with a learning rate of 0.0001 and an initial accumulator value of 0.1 was used. The maximum norm of gradient clipping was set to 2.0. We set the batch size to 8 on the CNN/Daily Mail dataset whereas 32 on the Newsroom dataset. Summaries were decoded through beam search with a beam size of 4 at test time. The maximum iterations on CNN/Daily Mail and Newsroom were 500,000 and 450,000, which are both approximately equal to 14 epochs. The same settings were applied to the baselines for comparison.

Training our embedding-level reconstruction model on the CNN/Daily Mail dataset took 41.6 hours, while it took 34.7 hours on the Newsroom dataset. We noticed that embedding-based approaches do not increase training time significantly compared to the baselines. Our approach can improve the performance without introducing too many parameters or sacrificing training efficiency.

When setting the scaling factor $\lambda$ (in Eq. 15), we found that the baselines with embedding-level reconstructors (in Sec. 3.2, 3.3) achieved the best results when the reconstruction loss was weighted to $\lambda = 2.0$ on the CNN/Daily Mail validation dataset and $\lambda = 2.5$ on the Newsroom validation dataset. However, the sequence-level reconstructor (in Sec. 3.1) worked best when $\lambda$ was set to 0.1.

Baselines  We employed the following excellent baselines for comparison and to demonstrate that our approaches can be transplanted to various Seq2Seq models. As our original intention to design the reconstructors, the PGN in Sec. 2 was treated as our main baseline for both datasets. We also adopted PGN+Coverage and PreSumm (Liu and Lapata, 2019)\(^1\) on the CNN/Daily Mail dataset to further examine the adaptability of our reconstructors. The coverage mechanism (Tu et al., 2016; See et al., 2017) maintains a vector, i.e., a sum of attention distributions over all the former decoding steps, to prevent repeating words. PreSumm employs and fine-tunes the pre-trained context representations of BERT (Devlin et al., 2018) as an encoder. On the Newsroom dataset, we additionally utilized LeafNATS (Shi et al., 2019)\(^2\) as our baseline. LeafNATS is an open-source toolkit that can train and evaluate neural Seq2Seq models for the abstractive summarization. The authors modified the PGN by adding an intra-decoder (Paulus et al., 2018) to it.

4.2 Results

CNN/Daily Mail  Table 3 shows our main results on the CNN/Daily Mail test set with ROUGE. Underlines indicate statistically significant differences from the baseline using the bootstrap test (Dror et al., 2018). The results for the Lead-3 baseline method are shown at the top, with excellent abstractive representatives in the middle, and our reconstruction methods at the bottom.

From Table 3, we can explicitly observe that the sequence-level reconstruction mechanism beats the baseline but took three times as long as the original model to train (117.9 hours), which is computationally expensive and time-consuming. Therefore, even though the sequence-level reconstruction has

\(^1\)https://github.com/nlpyang/PreSumm
\(^2\)https://github.com/tshi04/LeafNATS
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>Validation</th>
<th>Test</th>
<th>Article length</th>
<th>Summary length</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN/Daily Mail</td>
<td>287,226</td>
<td>13,368</td>
<td>11,490</td>
<td>781</td>
<td>56</td>
</tr>
<tr>
<td>Newsroom</td>
<td>993,101</td>
<td>108,621</td>
<td>108,670</td>
<td>751</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 2: Basic statistics of the datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead-3 Baseline (See et al., 2017)</td>
<td>40.34</td>
<td>17.70</td>
<td>36.57</td>
</tr>
<tr>
<td>Pointer-Generator (See et al., 2017)</td>
<td>36.44</td>
<td>15.66</td>
<td>33.42</td>
</tr>
<tr>
<td>Pointer-Generator + Coverage (See et al., 2017)</td>
<td>39.53</td>
<td>17.28</td>
<td>36.38</td>
</tr>
<tr>
<td>PreSumm (Liu and Lapata, 2019)</td>
<td>41.72</td>
<td>19.39</td>
<td>38.76</td>
</tr>
<tr>
<td>Pointer-Generator (our implementation)</td>
<td>36.69</td>
<td>15.90</td>
<td>33.45</td>
</tr>
<tr>
<td>+ Sequence-level reconstruction</td>
<td>37.10</td>
<td>16.32</td>
<td>33.92</td>
</tr>
<tr>
<td>+ Word embedding reconstruction</td>
<td>38.21</td>
<td>16.85</td>
<td>35.01</td>
</tr>
<tr>
<td>+ IDF-weighted embedding reconstruction</td>
<td>38.48</td>
<td>17.05</td>
<td>35.23</td>
</tr>
<tr>
<td>Pointer-Generator + Coverage (our implementation)</td>
<td>39.45</td>
<td>17.31</td>
<td>36.01</td>
</tr>
<tr>
<td>+ Word embedding reconstruction</td>
<td>40.02</td>
<td>17.91</td>
<td>36.74</td>
</tr>
<tr>
<td>+ IDF-weighted embedding reconstruction</td>
<td>40.40</td>
<td>18.21</td>
<td>37.12</td>
</tr>
<tr>
<td>PreSumm (rerun)</td>
<td>41.2</td>
<td>18.99</td>
<td>38.29</td>
</tr>
<tr>
<td>+ IDF-weighted embedding reconstruction</td>
<td><strong>41.55</strong></td>
<td><strong>19.17</strong></td>
<td><strong>38.56</strong></td>
</tr>
</tbody>
</table>

Table 3: ROUGE F1 scores on the test set of the CNN/Daily Mail dataset. The best results of our experiments are marked in bold. Underlined results significantly surpass the PGN, coverage or PreSumm baseline with $p < 0.01$.

been proved to work well for NMT (Tu et al., 2017), it is not suitable for the summarization task. One of the most likely explanations is that NMT is a sentence or document transformation between two languages, which attempts not to lose any information during the process. However, summarization is designed to compress the information to form a shorter version of the original article. The nature of this task makes it extremely difficult to reproduce the whole input sequence from the generated summary. Furthermore, recovering the entire original information is not necessary and does not match the summarization objective. Therefore, we report the results of the PGN-based sequence-level reconstructor only on this dataset.

For both the PGN and the Coverage baseline, we can see higher ROUGE scores achieved by the word embedding-level reconstructions, which demonstrates the their effectiveness. In addition, our third reconstructor with IDF-weighted embeddings outperformed the baselines and two other reconstruction methods, despite far fewer training epochs.

Even though we did not observe as much improvements as the previous two baselines with the PreSumm-based reconstruction, the statistical significance test shows the stability and effectiveness of our method. To overcome the appearance of rare words, PreSumm tokenize words into subwords with Byte Pair Encoding (Sennrich et al., 2016). However, one subword may appear in words with various IDF values, which makes it meaningless to calculate IDF in the granularity of subwords. Therefore, in our experiments, we still calculated IDF on the word-level while we gave the IDF weights for the words only to their first subword. We believe that the difference of the granularity between IDFs and embeddings is the main reason of the slight improvement. We leave how to solve this issue as our future work.

Newsroom Table 4 lists the comparison results on the Newsroom dataset. Following the previous one, we enumerated extractive methods, abstractive or mixed Seq2Seq models, and our reconstruction architectures in three blocks in turn. Obviously, our reconstruction-based models were largely superior to the extractive methods and two abstractive approaches, while achieving comparable ROUGE scores with ExtConSumm, which is the state-of-
<table>
<thead>
<tr>
<th>Method</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead-3 Baseline (Grusky et al., 2018)</td>
<td>32.02</td>
<td>21.08</td>
<td>29.59</td>
</tr>
<tr>
<td>TLM (Subramanian et al., 2019)</td>
<td>33.24</td>
<td>20.01</td>
<td>29.21</td>
</tr>
<tr>
<td>ExtConSumm Extractive (Mendes et al., 2019)</td>
<td>39.40</td>
<td>27.80</td>
<td>36.20</td>
</tr>
<tr>
<td>LeafNATS (Shi et al., 2019)</td>
<td>39.91</td>
<td>28.38</td>
<td>36.87</td>
</tr>
<tr>
<td>Pointer-Generator (our implementation)</td>
<td>37.12</td>
<td>25.27</td>
<td>33.92</td>
</tr>
<tr>
<td>+ Word embedding reconstruction</td>
<td>38.76</td>
<td>26.88</td>
<td>35.47</td>
</tr>
<tr>
<td>+ IDF-weighted embedding reconstruction</td>
<td>39.19</td>
<td>27.32</td>
<td>35.95</td>
</tr>
<tr>
<td>LeafNATS (rerun)</td>
<td>39.01</td>
<td>27.21</td>
<td>35.77</td>
</tr>
<tr>
<td>+ Word embedding reconstruction</td>
<td>39.36</td>
<td>27.49</td>
<td>36.15</td>
</tr>
<tr>
<td>+ IDF-weighted embedding reconstruction</td>
<td><strong>39.57</strong></td>
<td><strong>27.79</strong></td>
<td><strong>36.34</strong></td>
</tr>
</tbody>
</table>

Table 4: ROUGE F1 scores on the test set of the Newsroom dataset. The best results from our experiments are marked in bold. Underlined results significantly surpass the PGN or LeafNATS baseline with $p < 0.01$.

<table>
<thead>
<tr>
<th>Model</th>
<th>Informativeness</th>
<th>Readability</th>
<th>Redundancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGN</td>
<td>3.98</td>
<td>4.05</td>
<td>2.90</td>
</tr>
<tr>
<td>+ Embed. Rec.</td>
<td><strong>4.12</strong></td>
<td><strong>4.12</strong></td>
<td><strong>3.00</strong></td>
</tr>
<tr>
<td>+ IDF-wt. Rec.</td>
<td><strong>4.12</strong></td>
<td><strong>4.15</strong></td>
<td><strong>3.21</strong></td>
</tr>
</tbody>
</table>

Table 5: Human evaluation of model-generated summaries.

The art mixed method. The IDF-weighted word embedding reconstruction was still the best among the reconstructors, which achieved an average improvement of 2.05 ROUGE points over the PGN baseline.

To sum up, these results indicate that simply adding the IDF-weighted embedding-level reconstruction to a Seq2Seq model is a very useful method in abstractive document summarization. However, the training time for introducing the sequence-level reconstructor greatly increased while it gains only a small improvement compared with the other two effective word embedding-level reconstruction methods. We leave the problem of how to successfully reconstruct a long document from a short summary with a neural Seq2Seq model as future work.

5 Analysis

5.1 Human Evaluation

Next, we performed the experiments with a manual evaluation to investigate the quality of summaries in three aspects: informativeness, readability, and redundancy. We randomly selected 100 examples from each dataset. Forty volunteers on Amazon Mechanical Turk (AMT) with a U.S. high school diploma or higher qualification were asked to rate each summary on a scale of 1-5 (higher is better). The average scores for the summaries of each model are shown in Table 5.

As we can see, the baseline model suffered from low informativeness and high redundancy, which can be considered as under-attention and over-attention, respectively. Incorporating the reconstruction architectures could alleviate these problems, whereas better summaries were generated with the help of the IDF values. We consider two reasons for this, as follows. 1) When average word vectors of summaries differ from their source texts, the summaries tend to be penalized by the reconstruction loss. There was no big difference observed between using and not using the IDF weights in terms of informativeness because the word embeddings play an important role in covering all the input tokens. 2) The IDF value for each word serves as a discriminator for avoiding fair attention. The redundancy was reduced with the IDF weights because higher-valued words control the summary content while lower-valued words tend to be ignored.

5.2 Case Study

Table 1\(^3\) shows example summaries obtained by the PGN with and without the IDF-weighted embedding reconstruction. For ease of understanding, we marked four most important passages in the source article with different colors. By observing these example summaries, we identified the following issues: 1) The baseline model tended to incorrectly

\(^3\)Refer back to page 2.
focus on unimportant details in the original text, and 2) The whole sentences or paragraphs were frequently copied from the source even if half of them were meaningless or redundant. For example, the second sentence of the baseline summary indicates that the cause of the fire accident has not been investigated. Compared to other elements in the news, e.g., the incident, location, consequences, and solution, the cause is not essential and should not appear in the summary. Moreover, the redundancy can be reduced if the model ignores the attributive clause containing four alarms instead of copying the whole sentence from the article.

All of the four key messages were included in the summary generated by our IDF-weighted embedding reconstruction method, while only one of them appeared in the baseline. Even the reference misses one key piece of information. This example demonstrates the efficacy of our reconstruction mechanism in keeping the salient information in text summarization in addition to the improved ROUGE scores.

6 Related Work

Compared to the extractive summarization, abstractive methods are more challenging and attract attention because they can generate new words through the source document representation (Liddy, 2001; Nallapati et al., 2016). With the popularity of deep learning, many neural network-based models, especially Seq2Seq models (Rush et al., 2015; Chopra et al., 2016), have been widely applied to natural language processing tasks, such as machine translation (Bahdanau et al., 2015) and dialogue systems (Lei et al., 2018). Since the work of Rush et al. (2015), neural Seq2Seq networks with an attention mechanism have been widely utilized in the abstractive summarization tasks.

However, the attention mechanism is sometimes not enough to address different problems. For example, repetitions at the word or phrase level cause grammatical errors and insufficient reflection of the main idea of the source article. Therefore, the distraction method (Nema et al., 2017) imposes a constraint over the attention that can reduce the probability of repeated content. Tu et al. (2016) and See et al. (2017) found that the original attention often leads to over- or under-focus without the memory of past alignment information. Thus, they used the coverage concept from statistical machine translation to keep track of the attention history with an additional loss. Moreover, the inability to handle out-of-vocabulary (OOV) tokens also limits the fluency and readability of generated summaries. To alleviate this problem, hybrid models that combine the extractive and abstractive methods through the copy mechanism account for the vast majority of models used in the summarization task (Vinyals et al., 2015; Gu et al., 2016; See et al., 2017).

Pre-trained language models (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018), essentially word embeddings presenting contextual representations, that were learned from large-scale corpora, have recently emerged and achieved state-of-the-art performances in a variety of NLP tasks (Zhang et al., 2019; Liu and Lapata, 2019; Rothe et al., 2020). Due to the subword tokenizer, out-of-vocabulary words are rarely observed in their output even without the pointer mechanism.

However, neither non-pretrained Seq2Seq models nor BERT can measure the proportion of information transmitted from the source to the target. The idea of reconstruction can be implemented in many forms so as to adapt to different types of tasks. For example, Srivastava et al. (2015) proposed an LSTM encoder-decoder model that encodes the video and reconstructs its frame sequence. Tu et al. (2017) proposed a reconstruction model based on NMT that consists of three sequences. If the original sentence can be reconstructed from the target, it proves that the information has been effectively transferred. Another work is $SEQ^2$ (Baziotis et al., 2019) with a triple sequence structure, which rebuilds the input sentence from the latent representation of the decoder in the unsupervised sentence compression task.

7 Conclusion

In this work, we presented three reconstruction mechanisms for the neural Seq2Seq abstractive summarization task that reconstruct the essential information from the source document to its target summary. The proposed reconstruction methods are applicable to any attention-based Seq2Seq summarization architectures. Experimental results on both the CNN/Daily Mail and Newsroom datasets showed the improvements from the baselines in terms of ROUGE metrics and human evaluation. Our analysis also indicated that the proposed reconstruction approaches can restrict the under-attention to key points and over-attention to redundant parts.
References


Interpretable Propaganda Detection in News Articles

Seunghak Yu1∗ Giovanni Da San Martino2
Mitra Mohtarami3 James Glass3 Preslav Nakov4
1 Amazon Alexa AI, Seattle, WA, USA
2 Department of Mathematics, University of Padova, Italy
3 MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA
4 Qatar Computing Research Institute, HBKU, Qatar

yuseungh@amazon.com, dasan@math.unipd.it
{mitra, glass}@csail.mit.edu, pnakov@hbku.edu.qa

Abstract

Online users today are exposed to misleading and propagandistic news articles and media posts on a daily basis. To counter thus, a number of approaches have been designed aiming to achieve a healthier and safer online news and media consumption. Automatic systems are able to support humans in detecting such content; yet, a major impediment to their broad adoption is that besides being accurate, the decisions of such systems need also to be interpretable in order to be trusted and widely adopted by users. Since misleading and propagandistic content influences readers through the use of a number of deception techniques, we propose to detect and to show the use of such techniques as a way to offer interpretability. In particular, we define qualitatively descriptive features and we analyze their suitability for detecting deception techniques. We further show that our interpretable features can be easily combined with pre-trained language models, yielding state-of-the-art results.

1 Introduction

With the rise of the Internet and social media, there was also a rise of fake (Nguyen et al., 2020), biased (Baly et al., 2020a,b), hyperpartisan (Potthast et al., 2018), and propagandistic content (Da San Martino et al., 2019b). In 2016, news got weaponized, aiming to influence the US Presidential election and the Brexit referendum, making the general public concerned about the dangers of the proliferation of fake news (Howard and Kollanyi, 2016; Faris et al., 2017; Lazer et al., 2018; Vosoughi et al., 2018; Bovet and Makse, 2019).

There were two reasons for this. First, disinformation disguised as news created the illusion that the information is reliable, and thus people tended to lower their barrier of doubt compared to when information came from other types of sources.

Second, the rise of citizen journalism led to the proliferation of various online media, and the veracity of information became an issue. In practice, the effort required to fact-check the news, and its bias and propaganda remained the same or even got more complex, compared to traditional media, since the news was re-edited and passed through other media channels.

Propaganda aims to influence the audience with the aim of advancing a specific agenda (Da San Martino et al., 2020b). Detecting it is tricky and arguably more difficult than finding false information in an article. This is because propagandistic articles are not intended to simply make up a story with objective errors, but instead use a variety of techniques to convince people, such as selectively conveying facts or appealing to emotions (Jowett and O’Donnell, 2012).
While many techniques are ethically questionable, we can think of propaganda techniques as rhetorical expressions that effectively convey the author’s opinion (O’Shaughnessy, 2004). Due to these characteristics, propagandistic articles are often produced primarily for political purposes (but are also very common in commercial advertisement), which directly affect our lives, and are commonly found even in major news media outlets, which are generally considered credible.

The importance of detecting propaganda in the news has been recently emphasized, and research is being conducted from various perspectives (Rashkin et al., 2017; Barrón-Cedeno et al., 2019a; Da San Martino et al., 2019b). However, while previous work has done reasonable job at detecting propaganda, it has largely ignored the question of why the content is propagandistic, i.e., there is a lack of interpretability of the system decisions, and in many cases, there is a lack of interpretability of the model as well, i.e., it is hard to understand what the model actually does even for its creators.

Interpretability is indispensable if propaganda detection systems are to be trusted and accepted by the users. According to the confirmation bias theory (Nickerson, 1998), people easily accept new information that is consistent with their beliefs, but are less likely to do so when it contradicts what they already know. Thus, even if a model can correctly predict which news is propagandistic, if it fails to explain the reason for that, people are more likely to reject the results and to stick to what they want to believe. In order to address this issue, we propose a new formulation of the propaganda detection task and a model that can explain the prediction results. Figure 1 compares the coverage of the explanations for pre-existing methods vs. our proposal.

Our contributions can be summarized as follows:

- We study how a number of information sources relate to the presence and the absence of propaganda in a piece of text.
- Based on this, we propose a general framework for interpretable propaganda detection.
- We demonstrate that our framework is complementary to and can be combined with large-scale pre-trained transformers, yielding sizable improvements over the state of the art.

2 Task Setup

Given a document \( d \) that consists of \( n \) sentences \( d = \{d_i\}_{i=1}^n \), each sentence should be classified as belonging to one of 18 propaganda techniques or as being non-propaganda. The exact definition of propaganda can be subtly different depending on the social environment and the individual’s growth background, and thus it is not surprising that the propaganda techniques defined in the literature differ (Miller, 1939; Jowett and O’Donnell, 2012; Hobbs and McGee, 2014; Torok, 2015; Weston, 2018). The techniques we use in this paper are shown in Table 1. Da San Martino et al. (2019b) derived the propaganda techniques from the literature: they selected 18 techniques and manually annotated 451 news articles with a total of 20,110 sentences. This dataset\(^1\) has fragment-level labels that can span over multiple sentences and can overlap with other labeled spans.

This granular labeling went beyond our scope and we had to restructure the data. First, we divided the data into sentences. Second, in order to reduce the complexity of the task, we changed the multi-label setup to a multi-class one by ignoring duplicate labels and only allowing one technique per sentence (the first one), breaking ties at random. As a result, we obtained 20,111 sentences labeled with a non-propaganda class or with one of 18 propaganda techniques. Based on this data, we built a system for predicting the use of propaganda techniques at the sentence level, and we provided the semantic and the structural information related to propaganda techniques as the basis of the results.

3 Proposed Method

Our method can detect the propaganda for each sentence in a document, and can explain what propaganda technique was used with interpretable semantic and syntactic features. We further propose novel features conceived in the study of human behavioral characteristics. More detail below.

3.1 People Do Not Read Full Articles

Behavior studies have shown that people read less than 50% of the articles they find online, and often stop reading after the first few sentences, or even after the title (Manjoo, 2013). Indeed, we found that 77.5% of our articles use propaganda techniques in the first five sentences, 65% do so in the first three sentences, and 31.07% do so in the title.

\(^1\)http://propaganda.math.unipd.it/
We define the relative position of a sentence as $f_i$ with respect to the title (Fang et al., 2019) fine-tuned on the Fake News Challenge dataset (Hanselowski et al., 2018). The title of an article typically contains the topic and also the author’s view of that topic. Thus, we hypothesize that propaganda should also focus on the topic expressed in the title.

We represent the relationship between the target sentence and the title by measuring the semantic similarity $f_{i}^{sim}$ between them as the cosine between the sentence-BERT representations ($\phi(x)$) (Reimers and Gurevych, 2019) of the target sentence $d_i$ and of the title $d_t$.

$$f_{i}^{sim} = \frac{\phi(d_i) \cdot \phi(d_t)}{|\phi(d_i)||\phi(d_t)|} \quad (1)$$

We further model the stance of a target sentence with respect to the title $f_{i}^{sent}$ using a distribution over five classes: related, unrelated, agree, disagree, and discuss. For this, we use a BERT model (Fang et al., 2019) fine-tuned on the Fake News Challenge dataset (Hanselowski et al., 2018).

### Technques and Definitions

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name Calling</td>
<td>give an object an insulting label</td>
</tr>
<tr>
<td>Repetition</td>
<td>inject the same message over and over</td>
</tr>
<tr>
<td>Slogans</td>
<td>use a brief and memorable phrase</td>
</tr>
<tr>
<td>Appeal to Fear</td>
<td>plant fear against other alternatives</td>
</tr>
<tr>
<td>Doubt</td>
<td>questioning the credibility</td>
</tr>
<tr>
<td>Exaggeration</td>
<td>exaggerate or minimize something</td>
</tr>
<tr>
<td>Flag-Waving</td>
<td>appeal to emotions or stereotypes</td>
</tr>
<tr>
<td>LL</td>
<td>the disguised group likes the idea</td>
</tr>
<tr>
<td>RtoH</td>
<td>appeal to popularity</td>
</tr>
<tr>
<td>CO</td>
<td>assume a simple cause for the outcome</td>
</tr>
<tr>
<td>OIC</td>
<td>use obscure expressions to confuse</td>
</tr>
<tr>
<td>AA</td>
<td>use authority’s support as evidence</td>
</tr>
<tr>
<td>B&amp;W Fallacy</td>
<td>present only two options among many</td>
</tr>
<tr>
<td>TC</td>
<td>discourage meaningful discussion</td>
</tr>
<tr>
<td>Red Herrings</td>
<td>introduce irrelevant material to distract</td>
</tr>
<tr>
<td>Straw Men</td>
<td>refute a nonexistent argument</td>
</tr>
<tr>
<td>Whataboutism</td>
<td>discredit an opponent’s position</td>
</tr>
</tbody>
</table>

We used three types of features ($f_{i}^{dp}$, $f_{i}^{sent}$, $f_{i}^{doc}$) to account for these observations, which we describe below.

#### 3.1.1 Relative Position of the Sentence

We define the relative position of a sentence as $f_{i}^{dp} = i/n$, where $i$ is the sequence number of the sentence, and $n$ is the total number of sentences in the article.

#### 3.1.2 Topic Similarity and Stance with Respect to the Title

The title of an article typically contains the topic and also the author’s view of that topic. Thus, we hypothesize that propaganda should also focus on the topic expressed in the title.

We represent the relationship between the target sentence and the title by measuring the semantic similarity $f_{i}^{sim}$ between them as the cosine between the sentence-BERT representations ($\phi(x)$) (Reimers and Gurevych, 2019) of the target sentence $d_i$ and of the title $d_t$.

$$f_{i}^{sim} = \frac{\phi(d_i) \cdot \phi(d_t)}{|\phi(d_i)||\phi(d_t)|} \quad (1)$$

We further model the stance of a target sentence with respect to the title $f_{i}^{sent}$ using a distribution over five classes: related, unrelated, agree, disagree, and discuss. For this, we use a BERT model (Fang et al., 2019) fine-tuned on the Fake News Challenge dataset (Hanselowski et al., 2018).

#### 3.2 Syntactic and Semantic Information

Some propaganda techniques have specific structural or semantic characteristics. For example, Loaded Language can be configured to elicit an emotional response, usually using an emotional noun phrase. To model this, we define the following three features: $f_{i}^{dp}$, $f_{i}^{sent}$, and $f_{i}^{doc}$.

#### 3.2.1 Syntactic Information

We used a syntactic parser to extract structural features about the target sentence $f_{i}^{dp}$. Our hypothesis is that such information could help to discover techniques that have specific structural characteristics such as Doubt and Black and White Fallacy. We considered a total of 27 clause-level classes and phrase-level labels, including the unknown class. The set is shown in Table 2.

#### 3.2.2 Sentiment of the Sentence

The sentiment of the sentence $f_{i}^{sent}$ is another important feature for detecting propaganda. This is because many propagandistic articles try to convince the readers by appealing to their emotions and prejudices. Thus, we extract the sentiment using a sentiment analyzer trained on social media data (Hutto and Gilbert, 2014), which gives a probability distribution over the following three classes: positive, neutral, and negative. It further outputs compound, which is a one-dimensional normalized, weighted composite score. We use all four scores as features.

Table 2: The syntactic labels we used as features.

<table>
<thead>
<tr>
<th>Level</th>
<th>Phrases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clause</td>
<td>S, SBAR, SBARQ, SINV, SQ</td>
</tr>
<tr>
<td>Phrase</td>
<td>ADJP, ADVP, CONJP, FRAG, INTJ, LST, NAC, NP, NX, PP, PRN, PRT, QP, RRC, UCP, VP, WHADJP, WHAVP, WHADVP, WHNP, WHPP, X</td>
</tr>
</tbody>
</table>

The class unrelated indicates that the sentence is not related to the claim made in the title, while agree and disagree refer to the sentence agreeing/disagreeing with the title, and finally discuss is assigned when the topic is the same as that in the title, but there is no stance. We further introduce the related class as the union of agree, disagree, and discuss. We use as features the binary classification labels and also the probabilities for these five classes.
3.2.3 Document-Level Prediction

If the document is likely to be propagandistic, then each of its sentences is more likely to contain propaganda. To model this, we use as a feature \( f^{doc} \) the score of the document-level propaganda classifier Proppy (Barrón-Cedeno et al., 2019a). Note that Proppy is trained on articles labeled using media-level labels, i.e., using distant supervision. Therefore, all articles from a propagandistic source are considered to be propagandistic.

4 Experimental Results

In this section, we present our experimental setup for interpretable propaganda detection and the evaluation results from our experiments. Specifically, we perform three sets of experiments: (i) in Section 4.1, we quantitatively analyze the effectiveness of the features we proposed in Section 3; (ii) in Sections 4.2 and 4.3, we compare our feature-based model to the state-of-the-art model described in (Da San Martino et al., 2019b) using the experimental setup from that paper; (iii) in Section 4.4, we analyze the performance of our model with respect to each of the 18 propaganda techniques.

4.1 Quantitative Analysis of the Proposed Features

Figure 2 shows the absolute value of the covariance between each of our features \( f \) and each of the 18 propaganda techniques \( T \). We calculated the values of the features on the training and on the development datasets, and we standardized their values. Then, we formulated this as a problem of calculating the covariance between continuous and Bernoulli random variables as follows:

\[
\text{cov}(f, T) = p \cdot (1 - p) \cdot (E[f|T = 1] - E[f|T = 0]).
\]

The total number of sentences used is 16,137 (for the training and for the development datasets, combined), among which there are 4,584 propagandistic sentences. In Figure 2, the vertical axis represents the proposed features, and the horizontal axis shows the individual propaganda techniques and the total number of instances thereof. Each square shows an absolute value of the covariance between some feature and some propaganda technique. We show absolute values in order to ignore the direction of the relationship, and we apply a threshold of 0.001 in order to remove the negligible relations from the figure.

Although the most frequent propaganda techniques appear in less than 10% of the examples, they do show qualitatively meaningful associations. Indeed, we do not expect a feature to correlate with multiple techniques, as they are fundamentally different. We believe that having features that strongly correlate with one technique might be an advancement towards detecting that technique.

We can see that the structural information \( f^{dp} \) and the sentiment of a sentence \( f^{sent} \) are closely associated with certain propaganda techniques. For example, Loaded Language has a strong correlation with features identifying words bearing either a positive or a negative sentiment. This makes sense as the authors are more likely to use emotional words rather than neutral ones, and Loaded Language aims to elicit an emotional response. Similarly, Doubt has high correlation with certain syntactic categories.

There are a number of interesting observations about the other features. For example, the relative position of sentences \( f^{rp} \) is associated with more than half of the propaganda techniques. Moreover, the similarity to the title \( f^{sim} \) and the stance with respect to the title \( f^{sts} \) are strongly correlated with the likelihood that the target sentence is propagandistic. The features that indicate whether a sentence is related to the subject of the title are complementary, and thus the covariances are the same when absolute values are considered.

4.2 Comparison to Existing Approaches

Table 3 shows a performance comparison for our model vs. existing models on the sentence-level propaganda detection dataset (Da San Martino et al., 2019b). This dataset consists of 451 manually annotated articles, collected from various media sources, and a total of 20,111 sentences. Unlike the experimental setting in the previous sections, the task here is a binary classification one: given a sentence, the goal is to predict whether it contains at least one of the 18 techniques or not. For the performance comparison, we used BERT (Devlin et al., 2019), which we fine-tuned for sentence-level classification using the Multi-Granularity Network (MGN) (Da San Martino et al., 2019b) architecture on top of the [CLS] tokens (trained end-to-end), as this model improves the performance for both tasks by controlling the word-level prediction using information from the sentence-level prediction and vice versa.
Figure 2: Covariance matrix between the 18 propaganda techniques and the proposed features.
<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>fine-tuned BERT¹</td>
<td>63.20</td>
<td>53.16</td>
<td>57.74</td>
</tr>
<tr>
<td>MGN¹</td>
<td>60.41</td>
<td>61.58</td>
<td>60.98</td>
</tr>
<tr>
<td>Proposed</td>
<td>40.97</td>
<td>73.27</td>
<td>52.55</td>
</tr>
<tr>
<td>Proposed w/ emb</td>
<td>49.41</td>
<td>80.87</td>
<td>61.34</td>
</tr>
<tr>
<td>Proposed w/ emb - f_{stn}</td>
<td>49.59</td>
<td>81.44</td>
<td>61.64</td>
</tr>
</tbody>
</table>

Table 3: Comparison of our method to pre-existing propaganda detection models at the sentence level for binary classification (propaganda vs. non-propaganda). The models flagged with ¹ are described in (Da San Martino et al., 2019b).

<table>
<thead>
<tr>
<th>Ablations</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>40.97</td>
<td>73.27</td>
<td>52.55</td>
</tr>
<tr>
<td>- f_{frp}</td>
<td>40.87</td>
<td>73.17</td>
<td>52.45</td>
</tr>
<tr>
<td>- f_{sim}</td>
<td>40.85</td>
<td>70.87</td>
<td>51.83</td>
</tr>
<tr>
<td>- f_{stn}</td>
<td>40.07</td>
<td>69.62</td>
<td>50.86</td>
</tr>
<tr>
<td>- f_{dp}</td>
<td>37.85</td>
<td>61.54</td>
<td>46.87</td>
</tr>
<tr>
<td>- f_{sent}</td>
<td>30.53</td>
<td>77.69</td>
<td>43.83</td>
</tr>
</tbody>
</table>

Table 4: Ablation study for our model on binary propaganda detection at the sentence level.

4.3 Ablation Study

Next, we performed an ablation study of the binary (propaganda vs. non-propaganda) model discussed in Section 4.2. The results are presented in Table 4. The values in the last row of the table, i.e., \(- f_{sent}\), are obtained by applying the document-level classifier, i.e., the feature \(f_{doc}\), to all sentences. We can see that the structural information about the sentence \(f_{dp}\) is the best feature for this task. This is due to the nature of some propaganda techniques that must have a specific sentence structure, such as Doubt. In addition, as described above, since there are many techniques related to inducing emotional responses in the readers, it can be understood that the sentiment of a sentence may be a good feature, e.g., for Loaded Language. These results are consistent with our findings in Section 4.1 above. Moreover, the novel features we devised based on a human behavioral study for propaganda detection \(f_{frp}, f_{sim}, f_{stn}\) improved the performance further. Overall, we can see in the table that all features contributed to the performance improvement.

4.4 Detecting the 18 Propaganda Techniques

For the experiments described in the following, we revert back to the task formulation in Section 2, but we perform a more detailed analysis of the outcome of the model: for a given article, the system must predict whether each sentence uses propaganda techniques, and if so, which of the 18 techniques in Table 1 it uses.

Table 5 shows the performance of our model on this task. We can see in the rightmost column that some techniques appear only in a very limited number of examples, which explains the very low results for them, e.g., for Red Herring and Straw Man. In an attempt to counterbalance the lack of gold labels for some of the techniques, we used sentence embeddings with the proposed features to capture more semantic information. Since this task is more challenging than the binary classification problem, we can intuitively expect a performance reduction, resulting in a weighted average F1 score of 42.88. However, this formulation of the problem has the advantage of providing more granular predictions, thus enriching the propaganda detection results.

---

We followed the original data split when training and testing the model, which is 14,137/2,006/3,967 for training/development/testing. We trained a Support Vector Machine (SVM) model\(^2\) using the above-mentioned features and we optimized the values of the hyper-parameters on the development dataset using grid search. We used an RBF kernel with gamma=\{1e-3, 1e-4\} and C=\{10, 100\}.

We can see in Table 3 that our proposed model, which is based on interpretable features, performs relatively well when compared to fine-tuned BERT without direct semantic information about the target sentence. While our model is not state-of-the-art by itself, we managed to outperform the existing models and to improve over the state of the art by simply adding to it sentence embeddings as features (Reimers and Gurevych, 2019), which were not fine-tuned on propaganda data. However, when the stance of the sentence and the embedding of the sentence are used together, performance decreases. This may be due to the two techniques based on semantic similarity being somewhat inconsistent.

---

\(^2\)Ran on Intel Xeon E5-1620 CPU @ 3.60GHz x 4; 16GiB DDR3 RAM @ 1600MHz.
Table 5: Performance of our proposed method for the task of detecting the 18 propaganda techniques, as evaluated at the sentence level.

<table>
<thead>
<tr>
<th>Techniques</th>
<th>P</th>
<th>R</th>
<th>F1</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-propaganda</td>
<td>94.37</td>
<td>36.62</td>
<td>52.77</td>
<td>2,927</td>
</tr>
<tr>
<td>Name Calling</td>
<td>14.16</td>
<td>21.92</td>
<td>17.20</td>
<td>146</td>
</tr>
<tr>
<td>Repetition</td>
<td>4.60</td>
<td>5.59</td>
<td>5.05</td>
<td>143</td>
</tr>
<tr>
<td>Slogans</td>
<td>3.75</td>
<td>20.69</td>
<td>6.35</td>
<td>29</td>
</tr>
<tr>
<td>Appeal to F.</td>
<td>12.99</td>
<td>38.37</td>
<td>19.41</td>
<td>86</td>
</tr>
<tr>
<td>Doubt</td>
<td>5.97</td>
<td>34.85</td>
<td>10.20</td>
<td>66</td>
</tr>
<tr>
<td>Exaggeration</td>
<td>6.06</td>
<td>20.90</td>
<td>9.40</td>
<td>67</td>
</tr>
<tr>
<td>Flag-Waving</td>
<td>10.98</td>
<td>44.62</td>
<td>17.63</td>
<td>65</td>
</tr>
<tr>
<td>Loaded L.</td>
<td>32.80</td>
<td>20.13</td>
<td>24.95</td>
<td>303</td>
</tr>
<tr>
<td>Reduction</td>
<td>8.00</td>
<td>22.22</td>
<td>11.76</td>
<td>9</td>
</tr>
<tr>
<td>Bandwagon</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>3</td>
</tr>
<tr>
<td>Casual O.</td>
<td>4.03</td>
<td>27.27</td>
<td>7.02</td>
<td>22</td>
</tr>
<tr>
<td>O, I, C</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>5</td>
</tr>
<tr>
<td>Appeal to A.</td>
<td>1.32</td>
<td>13.04</td>
<td>2.39</td>
<td>22</td>
</tr>
<tr>
<td>B&amp;W fallacy</td>
<td>0.89</td>
<td>4.55</td>
<td>1.49</td>
<td>22</td>
</tr>
<tr>
<td>T. clichés</td>
<td>3.67</td>
<td>44.44</td>
<td>6.78</td>
<td>18</td>
</tr>
<tr>
<td>Red Herring</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>11</td>
</tr>
<tr>
<td>Straw Men</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
</tr>
<tr>
<td>Whataboutism</td>
<td>2.54</td>
<td>14.29</td>
<td>4.32</td>
<td>21</td>
</tr>
<tr>
<td>weighted avg</td>
<td>73.59</td>
<td>32.80</td>
<td>42.88</td>
<td>3,967</td>
</tr>
</tbody>
</table>

A more fine-grained propaganda analysis was proposed by Da San Martino et al. (2019b), who developed a corpus of news articles annotated with 18 propaganda techniques which was used in two shared tasks: at SemEval-2020 (Da San Martino et al., 2020a) and at NLP4IF-2020 (Da San Martino et al., 2019a). Subsequently, the Prta system was released (Da San Martino et al., 2020c), and improved models were proposed, addressing the limitations of transformers (Chernyavskiy et al., 2021). The Prta system was used to perform a study of COVID-19 disinformation and associated propaganda techniques in Bulgaria (Nakov et al., 2021a) and Qatar (Nakov et al., 2021b). Finally, multimodal content was explored in memes using 22 fine-grained propaganda techniques (Dimitrov et al., 2021a), which was also used in a SemEval-2021 shared task (Dimitrov et al., 2021b).

6 Conclusion and Future Work

We proposed a model for interpretable propaganda detection, which can explain which sentence in an input news article is propagandistic by pointing out the propaganda techniques used, and why the model has predicted it to be propagandistic. To this end, we devised novel features motivated by human behavior studies, quantitatively deduced the relationship between semantic or syntactic features and propaganda techniques, and selected the features that were important for detecting propaganda techniques. Finally, we showed that our proposed method can be combined with a pre-trained language model to yield new state-of-the-art results.

In future work, we plan to expand the dataset by creating a platform to guide annotators. The dataset will be updated continuously and released for research purposes. We also plan to release an interpretable online system, with the aim to foster a healthier and safer online news environment.

Acknowledgements

This research is part of the Tanbih mega-project, which aims to limit the impact of “fake news”, propaganda, and media bias by making users aware of what they are reading, thus promoting media literacy and critical thinking. It is developed in collaboration between the Qatar Computing Research Institute, HBKU and the MIT Computer Science and Artificial Intelligence Laboratory.

\[\text{http://propaganda.qcri.org/}\]
\[\text{http://tanbih.qcri.org/}\]
References


Philip N Howard and Bence Kollanyi. 2016. Bots, #StrongerIn, and #Brexit: Computational propaganda during the UK-EU referendum. Available at SSRN 2798311.


Farhad Manjoo. 2013. You won’t finish this article: Why people online don’t read to the end.

Clyde R Miller. 1939. The techniques of propaganda. From “how to detect and analyze propaganda,” an address given at town hall. The Center for learning.


Generic Mechanism for Reducing Repetitions in Encoder-Decoder Models

Ying Zhang¹, Hidetaka Kamigaito¹, Tatsuya Aoki¹, Hiroya Takamura² and Manabu Okumura¹

¹Tokyo Institute of Technology
²National Institute of Advanced Industrial Science and Technology (AIST)
{zhang,kamigaito,aoki,oku}@lr.pi.titech.ac.jp
takamura.hiroya@aist.go.jp

Abstract

Encoder-decoder models have been commonly used for many tasks such as machine translation and response generation. As previous research reported, these models suffer from generating redundant repetition. In this research, we propose a new mechanism for encoder-decoder models that estimates the semantic difference of a source sentence before and after being fed into the encoder-decoder model to capture the consistency between two sides. This mechanism helps reduce repeatedly generated tokens for a variety of tasks. Evaluation results on publicly available machine translation and response generation datasets demonstrate the effectiveness of our proposal.

1 Introduction

Sequence-to-sequence (seq2seq) models are a dominant paradigm in various natural language generation tasks, such as machine translation (Luong et al., 2015b; Tu et al., 2016), text summarization (Kiyono et al., 2018; Li et al., 2017), and response generation (Miller et al., 2017; Pasunuru and Bansal, 2018). As Mi et al. (2016) reported, however, basic seq2seq models (Bahdanau et al., 2015; Luong et al., 2015b) sometimes suffer from a repetition problem. One reason is that the attention mechanism does not explicitly consider which source side tokens have already been covered in the past attentions. As a result, the encoder repeatedly attends to the same token in the decoding steps, which leads to redundant generation.

Many researchers have proposed variants of the seq2seq model to tackle the problem. The coverage mechanism (Tu et al., 2016; Mi et al., 2016) prevented the model from generating redundant outputs by taking into account the coverage of the attention distribution. These approaches can be easily incorporated into the seq2seq model with only a single attention distribution between the encoder and the decoder. However, for seq2seq models with multiple attentions, such as Transformer (Vaswani et al., 2017), we cannot calculate the coverage of attentions, because the encoder attempts to attend to multiple attentions on each layer in the decoder. Thus, it is challenging to incorporate the coverage mechanism into the multi-attention based seq2seq models.

As another solution, Suzuki and Nagata (2017) proposed word-frequency estimation (WFE) that predicts the upper-bound frequency for each output token from the given input tokens to control redundancy in the output. Furthermore, Kiyono et al. (2018) proposed a source-side prediction module (SPM) that estimates the occurrences of input tokens from the hidden states of the decoder in the seq2seq model to reduce repetition. While WFE and SPM have an advantage in not depending on the structure of a seq2seq model, it is difficult to apply these approaches to some tasks other than text summarization because WFE and SPM assume that the input sentence contains more tokens than the output.

To cope with the above problems, in this work, we propose a generic approach for reducing the repetition, focusing on the differences between the embedding spaces of the source and target sides. Based on the assumption of distributional semantics, our approach regards the representations of an input sentence on both sides as word vectors, and attempts to minimize their difference during the training step. Hence, the seq2seq model can explicitly take into account the source side context also in the decoder.

Our experimental results on the IWSLT 2014 German-to-English translation task (Cettolo et al., 2014) and the PERSONA-CHAT response generation task (Zhang et al., 2018) showed that the proposed method effectively alleviates the repetition
problem for both tasks.

2 Seq2seq Model

Given a source sentence \( X = \{x_1, ..., x_I\} \), the seq2seq model generates a target sentence \( Y = \{y_1, ..., y_J\} \), where \( I \) and \( J \) are the numbers of source and target tokens, respectively. The seq2seq model consists of two main parts: encoder and decoder. The encoder computes the representation of a source sentence \( X \), and the decoder generates a target sentence by decomposing the conditional probability:

\[
p(Y|X) = \prod_{j=1}^{J} p(y_j|y_{<j}, X).
\]

(1)

3 Repetition Reduction Module

3.1 Overview

An overview of our proposed method, the repetition reduction module (RRM), is illustrated in Figure 1. We employ Transformer for both the encoder and decoder in the explanation in this section. Let \( \tilde{x} \) be the target side sentence representations of source sentence \( X \). With RRM, inspired by Kiyono et al. (2018) and Luong et al. (2015a), we consider \( \tilde{x} \) as the correct representation of \( X \) and try to reconstruct \( \tilde{x} \) in the target side. We use \( \tilde{q} \) to represent the reconstructed \( \tilde{x} \). Then the seq2seq model predicts not only the target side sequence \( Y \) but also \( \tilde{x} \). The prediction is written as follows:

\[
p(Y, \tilde{x}|X) = p(\tilde{x}|Y, X)p(Y|X).
\]

(2)

The conditional probability \( p(\tilde{x}|Y, X) \) has the role of preventing either over- or under-generation of \( Y \) by predicting the source side context until the decoding step ends. \( p(\tilde{x}|Y, X) \) can be simplified as \( p(\tilde{x}|X) \) if \( \tilde{q} \) does not depend on \( Y \). Since \( p(Y|X) \) is predicted by the seq2seq model as shown in Eq. (1), we give details of \( p(\tilde{x}|Y, X) \) in the next section.

3.2 Prediction of Source Side Context

Instead of using count-based discrete representations as in Kiyono et al. (2018), we incorporate continuous representations for both source and target sides to capture deeper semantic relations (Mikolov et al., 2013). We assume \( p(\tilde{x}|Y, X) \) is proportional to the similarity between the representations of the source sentence \( X \) before and after being encoded and decoded:

\[
p(\tilde{x}|Y, X) \propto \exp(\alpha(\cos(\tilde{x}, \tilde{q}))),
\]

(3)

where \( \alpha \) is a scaling factor.

Next, we explain the representations of the source sentence \( X \) in the source and target sides. Letting \( V_s \) be the source vocabulary, we define the indicator vector for the presence of source tokens as \( x_i \in \{0, 1\}^{\mid V_s \mid} \). The source side representation \( \tilde{x} \) of the source sentence is defined as follows:

\[
\tilde{x} = \sum_i E_{src} x_i,
\]

(4)

where \( E_{src} \in R^{H \times \mid V_s \mid} \) is a word embedding matrix for the source vocabulary, and \( H \) is the embedding size.

Similarly, we define the target side representation \( \tilde{q} \) of the source sentence as follows:

\[
\tilde{q} = \sum_j E_{src} q_j,
\]

(5)

where \( q_j \in R^{\mid V_s \mid} \) represents the probability distribution over the source vocabulary \( V_s \) at the \( j \)-th decoding step, which is calculated as follows:

\[
q_j = SoftMax(W_q \tilde{z}_j + b_q),
\]

(6)

where \( \tilde{z}_j \) is the final hidden state from the decoder, \( W_q \) is a weight matrix, and \( b_q \) is a bias term. Note that this softmax layer is only used in the training step.

3.3 Objective Function

By considering the negative log-likelihood of Eq. (2), we can induce our objective function \( G_t \) as follows:

\[
G_t = \sum_{(X,Y) \in D} \{ -\log p(Y|X) - \alpha(\cos(\tilde{x}, \tilde{q})) \},
\]

(7)

where \( D \) is a parallel training corpus.

4 Experiments

4.1 Experimental Settings

We first used the IWSLT 2014 German-to-English translation task to evaluate our method. The dataset is split into 160k/7k/7k sentences for training, validation, and test. Since Cho et al. (2014) reported that seq2seq models tend to produce few unknown tokens and yield high BLEU scores for short sentences in the neural machine translation task, we supposed longer sentences are vulnerable to be
over-translation, and our proposal would perform better for longer sentences. Therefore, we divided the test data into 3 parts: Short, Medium, and Long. In Short with 4927 pairs, the source contains no more than 25 byte pair encoding (BPE) (Sennrich et al., 2016) tokens. In Medium with 1524 pairs, the source contains 26 to 50 BPE tokens. In Long with 299 pairs, the source contains more than 50 BPE tokens.

We used PERSONA-CHAT for response generation as another dataset. This is the official dataset of The Conversational Intelligence Challenge 2 (ConvAI2) for testing chatbots. It contains 164k/15k/15k utterances (corresponding to 10k/1k/1k dialogs) for training, validation, and test. It also contains corresponding persona information for each dialog.

We used the model of Fonollosa et al. (2019) as a baseline for the machine translation task. And we regarded the best performing model (Wolf et al., 2019) in ConvAI2 as our baseline for the response generation task. Wolf et al. (2019) adopted a Generative Pretrained Transformer (Radford et al.) based encoder and a 12-layer Transformer decoder, and concatenated the persona information, up to two turns of history utterances, and the query (the utterance) together as an input sequence. To investigate the effectiveness of our proposed module, we compared the experimental results between the models with and without RRM on top of the baseline models.

For evaluation metrics, we used tokenized BLEU (Papineni et al., 2002), Meteor (Denkowski and Lavie, 2014), and Repeat (Kiyono et al., 2018) for the machine translation task. Repeat is defined as follows. Following the definition by Kiyono et al. (2018), we think that a model causes a repetition if it outputs the same token more than once. For each pair of a generated translation and its corresponding reference in the dataset, while we considered some tokens might occur more than once in the reference, Repeat was computed by subtracting the frequency of tokens in the reference from the frequency of tokens that occur more than once in the generated translation.

For the response generation task, we used the official evaluation metrics, F1 and Perplexity. Note that the official method, offered by ParlAI (Miller et al., 2017), ignores words \{a, an, the\} and punctuation when computing F1. To compute Perplexity, Wolf et al. (2019) indirectly predicted the word probability on the basis of the ratio of probabilities of subwords since they utilized a BPE vocabulary.

Different from the machine translation task, the response generation task has no fixed answer. Therefore, in this task, we ignored the reference sequence when we computed Repeat. For each generated sequence, Repeat was computed by subtracting 1 from the frequency of tokens that occur more than once in the generated sequence. While ignoring the words \{a, an, the\} and punctuation, we calculated Repeat scores under an n-gram setting at sentence-level and dialog-level. At the sentence-level, we calculated Repeat only with each generated response. At the dialog-level, we calculated Repeat with the concatenation of a sequence of the generated responses in a dialog.

For the machine translation task, we followed the experimental settings of Fonollosa et al. (2019),

---

\[^{1}\text{http://convai.io/}\]
and tuned the scaling factor $\alpha$ in Eq. (7) with Repeat as the evaluation metric on the validation dataset. For the response generation task, we carried on the experimental settings of Wolf et al. (2019), and tuned the scaling factor $\alpha$ with Repeat (Sentence-Level) under 1-gram as the evaluation metric on the validation dataset. See Appendix A for a complete list of hyperparameter settings.

### 4.2 German-to-English Results

Table 1 shows the experimental results for the German-to-English translation task. The results suggest that combining RRM with the model of Fonollosa et al. (2019) helps to improve Repeat, BLEU, and Meteor scores.

Next, we compared the experimental results for Short, Medium, and Long to investigate the effectiveness of RRM at different source sentence lengths. Table 2 summarizes the experimental results. Similar to the results from Cho et al. (2014), both models tended to have lower BLEU scores and more repetitions for longer sentences. RRM performed relatively well for longer sentences. It reduced more repetitions and improved more BLEU on top of Fonollosa et al. (2019) for longer sentences. While RRM ($\alpha = 0.3$) showed no effect in reducing the repetition for short sentences, it assisted the model of Fonollosa et al. (2019) in reducing the Repeat score by 0.335 points and improved BLEU by 0.25 points for long sentences. These results suggest the importance of RRM for long sentences.

In Table 3, we list top and bottom 20 words based on the degree of Repeat reduction by RRM ($\alpha = 0.3$). These results show that RRM tended to reduce repetitions for high frequency words. RRM showed no effect of reducing repetitions for “.” and “&apos;s.” See Appendix B for sample translations.

### 4.3 Response Generation Results

Tables 4 and 5 show the experimental results for the response generation task. Obviously, RRM reduced Repeat (Sentence-Level) by 0.056 (1-gram), and Repeat (Dialog-Level) by 0.471 (1-gram). The results suggest that combining RRM with the model of Wolf et al. (2019) helps to improve F1 and Repeat, while the performance of RRM in reducing perplexity is limited. We suppose that there are two reasons. One is that the probability calculation method offered by Wolf et al. (2019) is an indirect method. The other is that, the response is not fixed for a given source. See Appendix C for a sample dialog.

We conducted extensive experiments to investigate whether RRM has a potential to reduce more repetitions by considering the following conditions. First, beam search is an optimized decoding method which generates less repetitions than greedy decoding, and it may limit the performance of RRM. We investigate whether decoding methods might influence the performance of RRM by comparing beam search with greedy decoding.

Second, Wolf et al. (2019) used the persona information and the history utterances as an input sequence for their model to generate a response that differs from the history and contains a part of the persona information. However, in this task our model shared its vocabulary between source and target sides, and in Eq. (3) we utilized the cosine similarity to force $\mathbf{q}$ to be similar to $\mathbf{\tilde{x}}$, which may make the generated response similar to the input sequence. Therefore, we investigate whether using history utterances in the input sequence during testing might influence the performance of RRM. “w/o
<table>
<thead>
<tr>
<th>Word</th>
<th>Frequency</th>
<th>Frequency Rank</th>
<th>Sum of Repeat</th>
<th>Reduced Repeat</th>
</tr>
</thead>
<tbody>
<tr>
<td>you</td>
<td>45794</td>
<td>12</td>
<td>50</td>
<td>31</td>
</tr>
<tr>
<td>of</td>
<td>73774</td>
<td>6</td>
<td>76</td>
<td>62</td>
</tr>
<tr>
<td>to</td>
<td>67343</td>
<td>7</td>
<td>80</td>
<td>67</td>
</tr>
<tr>
<td>they</td>
<td>17469</td>
<td>27</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>and</td>
<td>96381</td>
<td>4</td>
<td>76</td>
<td>68</td>
</tr>
<tr>
<td>on</td>
<td>50883</td>
<td>10</td>
<td>56</td>
<td>48</td>
</tr>
<tr>
<td>do</td>
<td>11485</td>
<td>39</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>how</td>
<td>7222</td>
<td>59</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>to</td>
<td>71411</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>their</td>
<td>6187</td>
<td>17</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>to</td>
<td>134603</td>
<td>3</td>
<td>129</td>
<td>126</td>
</tr>
<tr>
<td>one</td>
<td>11115</td>
<td>40</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>would</td>
<td>6894</td>
<td>77</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>belief</td>
<td>120</td>
<td>1868</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>generally</td>
<td>101</td>
<td>2206</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>determine</td>
<td>86</td>
<td>2497</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>defined</td>
<td>85</td>
<td>2523</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>colleague</td>
<td>63</td>
<td>3210</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>eliminating</td>
<td>20</td>
<td>3737</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>celestial</td>
<td>15</td>
<td>8869</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>joints</td>
<td>15</td>
<td>8870</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>militarized</td>
<td>11</td>
<td>10831</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>anatomical</td>
<td>5</td>
<td>1781</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>humiliated</td>
<td>5</td>
<td>17812</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>for</td>
<td>18902</td>
<td>22</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>can</td>
<td>15244</td>
<td>29</td>
<td>11</td>
<td>14</td>
</tr>
<tr>
<td>people</td>
<td>10653</td>
<td>42</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>someone</td>
<td>695</td>
<td>415</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>river</td>
<td>146</td>
<td>1572</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>compromised</td>
<td>12</td>
<td>10263</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>had</td>
<td>6648</td>
<td>70</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>'</td>
<td>36045</td>
<td>15</td>
<td>39</td>
<td>47</td>
</tr>
<tr>
<td>'</td>
<td>191365</td>
<td>1</td>
<td>211</td>
<td>224</td>
</tr>
</tbody>
</table>

Table 3: Top and bottom 20 words based on the degree of Repeat reduction. They are listed in descending order of the Repeat reduction by +RRM ($\alpha = 0.3$) on top of Fonollosa et al. (2019) for the IWSLT 2014 De-En test dataset at Long length. Frequency is the frequency in the training dataset, and Frequency Rank is its rank in the training dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>1-gram</th>
<th>2-gram</th>
<th>3-gram</th>
<th>4-gram</th>
<th>5-gram</th>
<th>Perplexity</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wolf et al. (2019)*</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>16.28</td>
<td>19.50</td>
</tr>
<tr>
<td>Wolf et al. (2019)</td>
<td>0.755</td>
<td>0.244</td>
<td>0.107</td>
<td>0.056</td>
<td>0.025</td>
<td>16.31</td>
<td>18.22</td>
</tr>
<tr>
<td>+RRM</td>
<td>0.699</td>
<td>0.210</td>
<td>0.090</td>
<td>0.045</td>
<td>0.018</td>
<td>16.33</td>
<td>18.36</td>
</tr>
</tbody>
</table>

Table 4: Experimental results on the PERSONA-CHAT test dataset. **Bold** indicates the best scores. Results were the average over 3 runs by random seeds. * indicates the reported scores by Wolf et al. (2019). $\alpha$ was fixed to 0.3, that yielded the best with Repeat (Sentence-Level) under 1-gram on the validation dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>1-gram</th>
<th>2-gram</th>
<th>3-gram</th>
<th>4-gram</th>
<th>5-gram</th>
<th>Perplexity</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>+RRM</td>
<td>27.952</td>
<td>13.982</td>
<td>7.605</td>
<td>4.743</td>
<td>2.791</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Experimental results on the PERSONA-CHAT test dataset. **Bold** indicates the best scores. Results were the average over 3 runs by random seeds. $\alpha$ was fixed to 0.3, that yielded the best with Repeat (Sentence-Level) under 1-gram on the validation dataset.
to producing more repetitions at the sentence-level and hence more repetitions at the dialog-level. We therefore investigated whether using the persona information and the history utterances in Eq. (4) during training influences the performance of RRM. Full indicates the usage of the persona information, the history utterances and the query as a source in Eq. (4) during training, while part indicates the usage of only the query. We also tried the setting divide, which divides the input sequence in Eq. (4) into three parts, $\bar{x}_p, \bar{x}_h, \bar{x}_l$, depending on the persona information, the history utterances and the query, and uses the corresponding $W_{qp}, W_{qh}, W_{ql}$ in Eq. (6) to compute $q_p, \tilde{q}_h, \tilde{q}_l$ respectively. Then, the averaged cosine similarity was calculated between each divided $\bar{x}$ and $\tilde{q}$.

Tables 6 and 7 show the results of our extensive experiments. Clearly, RRM reduced more repetitions and improved F1 scores more when using beam search and the full setting. When the input sequence excluded history utterances during testing, RRM performed worse in F1 scores. Using only the query (part) in Eq. (4) during training reduced more repetitions at the sentence-level than using a full input sequence (full) when the input sequence excluded the history. But under other settings, RRM performed best when full was used. divide setting was the worst among the full, divide and part settings.

It indicates that our third supposition was wrong
and the part setting was unstable. We think the reason of such unstable performance is, when using the part setting, \( \tilde{q} \) and \( \tilde{x} \) in Eq. (3) were respectively generated from the full input sequence and only a part of the input sequence, which makes the information unbalanced.

The above results suggest that, when RRM was utilized, the method for combining multiple information for the input sequence was important. Furthermore, the decoding method would influence the performance of RRM.

5 Related Work

To overcome the repetition problem in neural machine translation, Tu et al. (2016) and Mi et al. (2016) introduced the coverage mechanism into a seq2seq model so that the decoder can pay attention to the encoder information without duplication. See et al. (2017) extended the coverage model by incorporating a pointer-generator network based on Tu et al. (2016). However, it is hard to utilize these coverage methods for multi-head attention based models because multi-head attention is a stack of several attention layers, and each layer is trained to capture its own distribution. Furthermore, the works of Tu et al. (2016) and Mi et al. (2016) are based on one-to-one correspondence generation, which cannot be applied to a “lossy” compression task such as summarization.

Suzuki and Nagata (2017) proposed word-frequency estimation (WFE) which used several linear transformations to map the hidden states of the encoder into the upper-bound occurrence of each target vocabulary and controlled the generation by the estimated occurrence. However, we cannot apply WFE for some generation tasks such as the response generation task, in which the frequency of target tokens is irrelevant to the source sentence. Kiyono et al. (2018) proposed a source-side prediction module (SPM) and assumed that output sentences are always shorter than input sentences (i.e., a summary or a headline of the input). To make sure the lengths of input and output sentences were equal, special \(< \text{pad} >\) tokens were added to the end of the target sentence. While this method helps SPM to estimate the over- or under-generation with the euclidean distance, it limits the application of SPM. Since our approach does not rely on the above assumptions, RRM is more scalable to other downstream tasks, including machine translation and response generation.

6 Conclusion

In this work, we proposed a novel mechanism to suppress repetitions in machine translation and response generation. Our model attempts to estimate the semantic vectors from a source sentence on both sides of an encoder-decoder model, which takes semantic repetitions into consideration and does not rely on any attention features. Therefore, it is potential to apply our proposal to other sequence-to-sequence models, which is an advantage of our approach compared with previous methods.

Experimental results on the IWSLT 2014 German-to-English translation task and the PERSONA-CHAT response generation task demonstrated the effectiveness of our proposal. The results of the extensive experiments in the response generation task showed RRM has the ability to handle a concatenated input sequence.

Because our proposal takes the semantic repetitions into consideration, we believe it might have the ability to reduce the repetitions among semantically similar words. We will verify it as future work.

References


A Hyperparameters

For the machine translation task, we followed the experimental settings of Fonollosa et al. (2019), which used fairseq (Ott et al., 2019) and a 31K-token BPE source and target vocabulary. For the hyperparameters of Transformer, we used 14 layers, an embedding size of 256, a feedforward expansion size of 1024, and 4 attention heads. We used the Adam (Kingma and Ba, 2015) optimizer with a 4k mini-batch size and 85k training steps. The learning rate was linearly warmed from $1 \times 10^{-7}$ to 0.001 in 4k steps and then decayed by a weight of 0.0001 (Loshchilov and Hutter, 2017). In the decoding steps, we used beam search (Wu et al., 2016) with a beam size of 5. We set the scaling factor $\alpha$ to \{1, 0.3, 0.2, 0.05, 0.01\} and selected the best $\alpha$ with Repeat as the evaluation metric on the validation dataset. We pretrained the model of Fonollosa et al. (2019) in advance to extract word embeddings.

For the response generation task, we carried on the experimental settings of Wolf et al. (2019). We utilized a 40k-token BPE source and target vocabulary and trained the model with 2 epochs, a batch size of 32 sequences, and the Adam optimizer. The learning rate was linearly decayed from $6.25 \times 10^{-5}$ to zero. In the decoding step, we adopted beam search, and top 20 sampling (Fan et al., 2018) was utilized before selecting four beams. We set the scaling factor $\alpha$ to \{1, 0.3, 0.2, 0.05, 0.01\}. Since RRM was designed to reduce repetitions at the sentence-level, we selected the best $\alpha$ with Repeat (Sentence-Level) under 1-gram as the evaluation metric on the validation dataset.

B Sample of German-to-English

Table 8 shows sample translations. While the model of Fonollosa et al. (2019) tended to generate repeating phrases, our model reduced such generation.

C Sample of Response Generation

We show a sample dialog in Table 9. Similar to the machine translation task, the model of Wolf et al. (2019) tended to generate repeating phrases, and our model helped to alleviate it. In particular, “i’m a real estate agent” in the second turn is a 5-gram repetition of the one in the first turn at the dialog-level when word “a” is ignored. “they are twins” in the third turn is a 3-gram repetition at both dialog-level and sentence-level because it is generated twice in a response.
the only real choice was who, not when, and not what you did after.

so we take something very complicated, we turn it into sound, a sequence of sound, and we produce something very complicated in the head of others.

it &apos;s a process, and sometimes it works and sometimes it doesn &apos;t. but the idea that we should not allow science to do its job because we &apos;re afraid, is really very deadening, and it &apos; s preventing millions of people from prospering.

Table 8: Sample translations for Short, Medium, and Long data. Underline indicates repetitions that contain more than two words, and bold indicates wrong translations. \( \alpha \) was fixed to 0.3, that yielded the best with Repeat on the validation dataset.

Table 9: Sample responses generated by various models. Underline indicates repetitions that contain more than two words, and bold indicates their counterparts. A turn is a pair of a query (an utterance) and its response (either a reference or a generation). In this example, a sequence of the three turns consist of a part of a dialog. When a model tries to output a response to the third query, the first and second turns are the history utterances, and it receives the concatenation of the persona information, the history utterances and the third query as an input. \( \alpha \) was fixed to 0.3, that yielded the best with Repeat (Sentence-Level) under 1-gram on the validation dataset.
Knowledge Distillation with BERT for Image Tag-Based Privacy Prediction

Chenye Zhao
University of Illinois at Chicago
Chicago, USA
czhao43@uic.edu

Cornelia Caragea
University of Illinois at Chicago
Chicago, USA
cornelia@uic.edu

Abstract

Text in the form of tags associated with online images is often informative for predicting private or sensitive content from images. When using privacy prediction systems running on social networking sites that decide whether each uploaded image should get posted or be protected, users may be reluctant to share real images that may reveal their identity, but may share image tags. In such cases, privacy-aware tags become good indicators of image privacy and can be utilized to generate privacy decisions. In this paper, our aim is to learn tag representations for images to improve tag-based image privacy prediction. To achieve this, we explore self-distillation with BERT, in which we utilize knowledge in the form of soft probability distributions (soft labels) from the teacher model to help with the training of the student model. Our approach effectively learns better tag representations with improved performance on private image identification and outperforms state-of-the-art models for this task. Moreover, we utilize the idea of knowledge distillation to improve tag representations in a semi-supervised learning task. Our semi-supervised approach with only 20% of annotated data achieves similar performance compared with its supervised learning counterpart. Last, we provide a comprehensive analysis to get a better understanding of our approach.

1 Introduction

With the rapid growth of the number of users on online social networking sites, image privacy has become a major concern (Ahern et al., 2007; Squicciarini et al., 2017). Users may accidentally disclose their sensitive information such as locations, habits or personal relationships from images that they post to their social networking sites (Squicciarini et al., 2017), which could be used in the detriment of the users (Tonge and Caragea, 2020). Zerr et al. (2012a) defines private images as ones that belong to the private sphere (e.g., portraits, family, home) or capture sensitive contents that can not be shared with everyone on the Internet. The remaining images are considered to be public. Binary image privacy classifiers are developed (Tonge and Caragea, 2018; Yang et al., 2020; Zerr et al., 2012a) aiming to identify whether images belong to the public class or the private class. However, the access to the image content is not always allowed since users may be reluctant to share the real images (revealing user’s identity through the face, and friends, etc.) for visual content analysis. In such cases, tags attached by users to describe their images are found to be informative about the image contents and are good indicators of the privacy settings and improve the privacy prediction methods (Tonge et al., 2018). Privacy prediction models trained with image tags achieve competitive results compared with vision-based privacy prediction models (Squicciarini et al., 2017). Therefore, our goal is to learn good tag representations for images to further improve the performance of tag-based privacy prediction.

Pre-trained language models have been extensively studied in NLP communities (Howard and Ruder, 2018; Devlin et al., 2019; Peters et al., 2018; Liu et al., 2019). BERT (Devlin et al., 2019) is a pre-trained language model based on a multi-layer bidirectional Transformer, and has shown to be effective for generating universal language representations and attains state-of-the-art performance on many natural language processing tasks (Dai and Callan, 2019; Adhikari et al., 2019). In our work, we fine-tune BERT for the task of tag-based image privacy prediction to generate better tag representations. In addition, we propose to use self knowledge distillation with BERT (Hinton et al., 2015; Clark et al., 2019; Zhang and Sabuncu, 2020) to further improve the performance of tag-based
privacy prediction. Specifically, we first train a BERT (Devlin et al., 2019) teacher model for privacy prediction, and then train a BERT student model using both class labels and the pre-trained teacher model’s output probability distributions. The student model can thus learn from not only the ground truth class information, but also how to assign compatible probabilities according to various input examples. Experimental results show that knowledge distillation effectively improves tag representations and achieves boosted prediction performance.

Moreover, training a classifier often requires a large amount of annotated data. However, the annotation process is very time consuming and requires a significant human effort in many cases (Deng et al., 2009). Thus, we investigate knowledge distillation in a semi-supervised learning approach (Xie et al., 2020). To do this, we first train a BERT teacher model using limited amount of labeled image tags, and use it to annotate a large amount of unlabeled data, which is further used to train another BERT student model for privacy prediction. Experimental results show that our semi-supervised approach with BERT learns good tag representations and achieves comparable performance with its supervised counterpart with only 20% annotated image tags.

Last, we provide a comprehensive analysis for our tag-based privacy prediction. First, we perform a calibration analysis to show that models trained by improved tag representations with knowledge distillation yield better calibration (the alignment between prediction confidence and correctness likelihood (Guo et al., 2017)). Second, neural models are sensitive to small perturbation in the input and a small perturbation on the input may fool a well-trained neural network (Hsieh et al., 2019; Belinkov and Bisk, 2018; Niu et al., 2020). We analyze the robustness of our privacy classification models trained by tag representations learned with knowledge distillation against adversarial attacks. The results show that our approach shows the most robustness against adversarial attacks over compared baselines. Third, we perform a statistical analysis on the correlation between the privacy and sentiment and emotion of image tags.

2 Related Works

Knowledge distillation. Knowledge distillation is originally proposed as a model compression method (Buciluă et al., 2006; Hinton et al., 2015). The standard knowledge distillation scheme transfers knowledge from a larger pre-trained “teacher” model to a smaller “student” model by training the student to mimic the class probability distributions generated by the teacher (Hinton et al., 2015). Recently, other works propose self-distillation (Furlanello et al., 2018; Clark et al., 2019): the teacher and the student have identical architectures, which achieve remarkable improvement on the student over the teacher. Zhang and Sabuncu (2020) experimentally demonstrate that the improvement of knowledge-distillation is correlated to the instance-level regularization on the student’s softmax outputs, meaning that by mimicking teacher’s probability distributions, instead of simply being trained to mimic one-hot class labels, the student are trained to assign compatible confidence (probabilities) according to the corresponding input examples. Meanwhile, one interesting focus of research on knowledge distillation has been on finding new applications. Chen et al. (2020) use the idea of knowledge distillation on BERT for text generation. Kim and Rush (2016) introduce knowledge distillation for sequence modeling. In contrast, we propose to utilize knowledge distillation as a tool to learn better tag representations for online images and to achieve improved performance for tag-based image privacy predictions.

Image privacy prediction. Most machine learning-based image privacy prediction models utilize images to train vision-based classification models to detect image privacy (Tran et al., 2016; Yang et al., 2020; Zerr et al., 2012b; Buschek et al., 2015). There are few works that adopt tags attached to describe images as indicators of image privacy and achieve competitive performance compared with vision-based approaches (Squicciarini et al., 2017). Further developing tag-based image privacy prediction approaches becomes a crucial direction for this task. Tonge and Caragea (2020) introduce TagCNN model based on the sentence classification CNN model (Kim, 2014) for image privacy prediction, where Word2Vec (Mikolov et al., 2013) is applied as the word embedding, and the CNN classifier is trained to predict image privacy. The bag-of-tags(BoT) model is introduced in (Tonge and Caragea, 2020) as another tag-based privacy prediction approach, where tags are embedded into multi-hot vectors similar to the bag-of-words embedding. Then a SVM classifier is trained for privacy detec-
tion. Previous tag-based works are trained using only the class labels. In contrast, we distill knowledge using BERT to utilize both hard class labels and soft probability distributions to improve tag representations and boost the performance of this task.

3 Methods

In this work, we adopt knowledge distillation with BERT to learn better tag representations for tag-based image privacy prediction. The idea behind knowledge distillation is that: soft probability distributions generated by a pre-trained image privacy prediction model carries additional privacy information compared with hard class labels. Specifically, hard labels can only reflect the class information (either private or public) of input image tags, while soft probability distributions can further reveal the confidence of the privacy classification model toward each prediction. A proper usage of such additional information, in combination with hard labels, can help learn better tag representations to boost the performance of tag-based image privacy prediction models. The goal of this work is to distill knowledge using BERT to transfer knowledge (in the form of soft probability distributions) from a strong, pre-trained BERT teacher model to a BERT student model to boost the performance of the latter for tag-based privacy prediction.

3.1 Knowledge Distillation with BERT

We first fine-tune a BERT (Devlin et al., 2019) as the teacher model for tag-based image privacy prediction. Given input image tags (x) for an online image, the teacher model generates a vector of scores, which is normalized to be the probability distribution of the two privacy classes: \( P_T = \text{softmax}(p_{\text{pub}}^T(x), p_{\text{pri}}^T(x)) \). As the first teacher model is trained using hard-labels, we adopt an temperature term \( T \) (Hinton et al., 2015) to ”soften” the probability distribution and avoid generating peaky probabilities: \( P_T = \text{softmax}(p_{T_{\text{pub}}}^T(x), p_{T_{\text{pri}}}^T(x))/T \). The teacher model is then trained using the cross-entropy loss:

\[
L_T = \text{CrossEntropy}(P_T, y) \tag{1}
\]

After that, we perform knowledge distillation from the trained teacher model to the student model. As shown in Figure 1, our goal is to teach the student model BERT_{student} to learn from both soft probabilities (soft labels) generated by the trained teacher model, and the class labels (hard labels). Therefore in the total loss function of the student model, we need to minimize the difference between the student’s predictions with both the ground truth hard label and the teacher’s predictions. BERT_{teacher} generates probability distributions \( P_T \) for input image tags x. The probability distribution generated by the student model BERT_{student} is denoted as \( P_S \). The training loss of the student model is the combination of the loss with the soft label \( P_T (L_{soft}) \) and the loss with the class label y \( (L_{hard}) \), where we use cross-entropy as loss functions. The above process can be denoted as:

\[
L_{soft} = \text{CrossEntropy}(P_S, P_T) \tag{2}
\]

\[
L_{hard} = \text{CrossEntropy}(P_S, y) \tag{3}
\]

\[
L_S = \alpha * L_{soft} + \beta * L_{hard} \tag{4}
\]

where \( \alpha \) and \( \beta \) are hyperparameters.

3.2 Semi-Supervised Learning Approach with BERT

While using more labeled data improves performance, manually annotating privacy of online images is very time consuming and requires human intensive effort. This motivates us to distill knowledge with BERT in a semi-supervised manner, where a BERT teacher model is first trained using a small portion of labeled data. The trained teacher is used to annotate the large portion of unlabeled data. Next, we integrate the data annotated by the trained teacher model and the originally labeled data as the overall training set to train a student model. The trained student model becomes the next teacher model and repeats the process.

We used 50\% of the whole training set \( D \) as the unlabeled set \( U \) and the rest set \( L \) is used to sample different fractions to be used as labeled data. In each experiment we randomly select \( l = L * k' = D * k \), a subset of \( L \), as the selected labeled set, which is used to train first teacher model. \( k = 0.5 * k' \) is a fraction parameter ranging from \([0\%, 50\%]\). Our semi-supervised learning process can be concluded as follows:

1. Train the initial teacher model \( T_0 \) with the selected labeled set \( l \).

2. Use the trained teacher model to annotate the unlabeled set \( U \). Then integrate \( l \) and
the annotated $U$ as the annotated training set: $A = I \cup U$.

3. Train the student model $S$ using $A$.

4. The student model $S$ becomes the new teacher model $T$. Go back to step (2).

4 Experiments and Results

4.1 Experimental Settings

**Dataset.** In this work, we use a dataset of 32,000 examples from PicAlert (Zerr et al., 2012a), which, to our knowledge, is the only publicly available dataset for online image privacy prediction that captures real privacy needs of current social networks’ users. Images and tags of PicAlert are crawled from Flickr and manually annotated by external viewers, who are instructed to mark images as “private” or “public” following the guidance: private images are defined as images belonging to private sphere or ones you do not want to share with everyone, and the rest are public (Zerr et al., 2012a). The dataset is randomly split into train set (22000), validation set (5000) and test set (5000). The public and private images are in the ratio of 3:1 in each set. Each experiment is repeated five times using five train/validation/test splits and averaged as the final result. We delete special characters in user tags and replace tags with occurrences lower than 2 with the keyword “(UNKNOWN)” as they may bring noises to the classification model (Tonge and Caragea, 2020).

**Model Configuration.** All models are implemented based on python 3.6 and Pytorch 1.3.1. For baseline models BoT and TagCNN, we apply the same hyper-parameters and network architectures suggested in (Tonge and Caragea, 2020). For BoT, we create a vector with the dimension of the vocabulary size and set 1 to the position of tags that exist in the image, and 0 otherwise (Tonge and Caragea, 2020). We fine-tune BERT with a learning rate of $2e^{-6}$ and the training batch size of 8. Hyper-parameters of BERT are selected on the validation set. We experiment different values weighting parameter $(\alpha, \beta)$ in Equation 4 and use $(0.7,0.3)$ for BERT$_{KD}$ as it shows the best performance. We also experiment with dynamically increase/decrease $\alpha$ and $\beta$ along with the training process but they do not show better performance.

**Research Questions.** In our work, we aim to validate tag representations learned by distilling knowledge with BERT for tag-based image privacy prediction. We address the following research questions:

1. How does the performance of the BERT-based knowledge distillation approach compared with state-of-the-art models for supervised tag-based image privacy prediction?

2. What is the performance of our semi-supervised learning approach and how does it compare with its supervised learning counterparts?

3. Whether tag representations learned with knowledge distillation yield better calibration (the alignment between prediction confidence and correctness likelihood) of models?
Table 1: Results of knowledge distillation with BERT and state-of-the-art models (with standard deviation) for supervised tag-based privacy prediction.

<table>
<thead>
<tr>
<th>Model</th>
<th>$F_{1\text{private}}$</th>
<th>$F_{1\text{public}}$</th>
<th>$F_{1\text{overall}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoT</td>
<td>0.613 ±0.02</td>
<td>0.901 ±0.004</td>
<td>0.831 ±0.005</td>
</tr>
<tr>
<td>TagCNN</td>
<td>0.629 ±0.02</td>
<td>0.903 ±0.005</td>
<td>0.839 ±0.008</td>
</tr>
<tr>
<td>BERT</td>
<td>0.664 ±0.014</td>
<td>0.906 ±0.003</td>
<td>0.849 ±0.005</td>
</tr>
<tr>
<td>TagCNN$_{KD}$</td>
<td>0.654 ±0.017</td>
<td>0.906 ±0.004</td>
<td>0.847 ±0.007</td>
</tr>
<tr>
<td>BERT$_{KD}$</td>
<td>0.681 ±0.01</td>
<td>0.907 ±0.003</td>
<td>0.855 ±0.005</td>
</tr>
</tbody>
</table>

4. Can tag representations learned with knowledge distillation show stronger robustness against some adversarial attacks toward image tags?

5. How does the privacy of images tags correlate with tag emotions?

4.2 Experimental Results

In this section, we discuss the experimental results designed to address the research questions. We measure the performance using the $F_1$-score for each class as well as the weighted average $F_1$-score over both classes (private and public).

4.2.1 Research Question 1

Table 1 shows the performance of the knowledge distillation approach and the state-of-art tag-based image privacy prediction models. To fine-tune BERT, we add a fully connected layer after the [CLS] token of the last BERT layer and fine-tune the whole network. For knowledge distillation, we use BERT as the teacher model to transfer knowledge to another BERT (denoted as BERT$_{KD}$) and TagCNN (denoted as TagCNN$_{KD}$), respectively. We observe that BERT$_{KD}$ outperforms the state-of-the-art models on every compared metric especially on $F_{1\text{private}}$, yielding a large improvement upto 6.8%. We also notice that both knowledge distillation approaches effectively improve the $F_{1\text{private}}$ of corresponding student models. For example, TagCNN$_{KD}$ improves TagCNN by 2.5%. BERT$_{KD}$ boosts $F_{1\text{private}}$ of BERT by 1.7%, which further pushes BERT$_{KD}$ to be the new state-of-the-art model for tag-based image privacy prediction. Such results suggest the effectiveness of our knowledge distillation approach of bringing the knowledge of the soft probability distributions generated by the teacher model to the training process of the student and achieves boosted performance. Note that, TagCNN$_{KD}$ does not outperform its teacher, i.e., BERT. Which suggests that in our task, a more compact student model may not always outperform its teacher. Standard deviation results show that BERT-based approaches are more stable on compared metrics. Knowledge distillation can help generate more stable results.

4.2.2 Research Question 2

In our semi-supervised learning experiment, we use BERT as the initial teacher model and another BERT as the student model, denoted as BERT$_{KD}$. We consider a baseline that use TagCNN as student model, denoted as TagCNN$_{semi}^{KD}$. Moreover, we also experiment with TagCNN playing the role of both $T_0$ and $S$, denoted as TagCNN$_{base}$. Experiments are performed using different amount of labeled data $L$ (controlled by the percentage parameter $\alpha$). We randomly select $\alpha$ ranging from 0.25% to 100% of labeled data $L$. We repeat the student-teacher rotation 3 times and report the $F_{1\text{private}}$ of the student model in the last iteration. Results are shown in Figure 2, where we observe some trends of $F_{1\text{private}}$. Firstly, BERT$_{semi}^{KD}$ consistently show significant improvements over the two baseline approaches, yielding improvements upto 19.6%. Secondly, BERT$_{semi}^{KD}$ at $\alpha = 20\%$ achieves comparable performance with its supervised learning counterpart BERT$_{KD}$ trained with 100% labeled data, with only a small under-performance of 0.6%. In contrast, TagCNN$_{semi}^{KD}$ and TagCNN$_{semi-base}^{KD}$ always perform much worse than their supervised learning counterparts. This encouraging result illustrates that our semi-supervised knowledge distillation approach with BERT can still be serviceable even
Table 2: Calibration results for TagCNN, TagCNN with knowledge distillation (CNN\textsubscript{KD}), BERT, and BERT with knowledge distillation (BERT\textsubscript{KD}). TS represents the results after using temperature scaling.

<table>
<thead>
<tr>
<th>Model</th>
<th>CNN</th>
<th>CNN\textsubscript{KD}</th>
<th>BERT</th>
<th>BERT\textsubscript{KD}</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECE</td>
<td>2.30</td>
<td>2.19</td>
<td>11.07</td>
<td>8.45</td>
</tr>
<tr>
<td>ECE+TS</td>
<td>1.25</td>
<td>0.91</td>
<td>4.39</td>
<td>2.46</td>
</tr>
</tbody>
</table>

Table 3: Results on knowledge distillation approaches with compared baselines against adversarial attacks for image tags.

<table>
<thead>
<tr>
<th>Model</th>
<th>$F^1_{\text{private}}$</th>
<th>$F^1_{\text{public}}$</th>
<th>$F^1_{\text{overall}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Random Attack</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TagCNN</td>
<td>0.958</td>
<td>0.987</td>
<td>0.980</td>
</tr>
<tr>
<td>BERT</td>
<td>0.976</td>
<td>0.992</td>
<td>0.988</td>
</tr>
<tr>
<td>TagCNN\textsubscript{KD}</td>
<td>0.966</td>
<td>0.988</td>
<td>0.983</td>
</tr>
<tr>
<td>BERT\textsubscript{KD}</td>
<td>0.996</td>
<td>0.999</td>
<td>0.998</td>
</tr>
<tr>
<td><strong>Synonym-based Attack</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TagCNN</td>
<td>0.875</td>
<td>0.961</td>
<td>0.939</td>
</tr>
<tr>
<td>BERT</td>
<td>0.939</td>
<td>0.980</td>
<td>0.970</td>
</tr>
<tr>
<td>TagCNN\textsubscript{KD}</td>
<td>0.902</td>
<td>0.966</td>
<td>0.950</td>
</tr>
<tr>
<td>BERT\textsubscript{KD}</td>
<td>0.962</td>
<td>0.986</td>
<td>0.980</td>
</tr>
</tbody>
</table>

4.2.3 Research Question 3

A well-calibrated classification model should be able to generate the probability of the predicted privacy class label (the confidence) which reflects its correctness likelihood (the accuracy) (Guo et al., 2017). In other words, a well-calibrated model should not only generate accurate predictions of image privacy, but should also “know what it does not know”, meaning that the model does not generate overly confident yet incorrect predictions. This is especially important for the task of tag-based image privacy prediction. For a privacy decision making system, so that if input image tags are misclassified to the wrong privacy class but with lower confidence, the system can pass the input example to the owner of the image to double-check as the privacy prediction model is not confident about its prediction (the privacy classification model “knows what it does not know”).

In this work, we first study the calibration of tag representations learned with TagCNN, TagCNN\textsubscript{KD}, BERT, and BERT\textsubscript{KD}. We then improve the calibration of compared models by performing post-hoc calibration for the predicted probabilities. Specifically, we adopt the temperature scaling (Desai and Durrett, 2020; Guo et al., 2017) to post-process model probabilities, where logits generated by compared models are divided by a temperature scaling term $T'$, which is optimized with respect to the cross-entropy loss on the validation set. Note that temperature scaling does not affect the model’s accuracy. To evaluate the calibration of model predictions, we use the expected calibration error (ECE), which is defined as the weighted average of the difference between accuracy and confidence in $m$ equally-partitioned confidence bins (Guo et al., 2017), where $m$ is commonly selected to be 10. Results are shown in Table 2, where we observe that tag representations learned with knowledge distillation improves model calibration for both TagCNN and BERT: ECE scores are reduced by upto 2.62%, suggesting that soft-labels from the teacher model alleviate the overconfident issue caused by hard labels. Interestingly, we also notice that BERT exhibits higher ECE than TagCNN. This is because BERT has much higher learning capacity than TagCNN. During training, after BERT is trained to correctly classify almost all training samples, the model is able to further increases its confidence towards predictions to achieve lower training loss, while TagCNN can not perform such further optimization due to its limited learning capacity. Thus BERT achieves better prediction accuracy, but result in larger ECE (Guo et al., 2017) compared with TagCNN. We also observe that temperature scaling effectively calibrates both BERT and TagCNN with significantly reduced ECE (Guo et al., 2017).

We also plot the reliability diagrams (Guo et al., 2017; Desai and Durrett, 2020) in Figure 3 to better visualize the alignment between the accuracy and the confidence of compared models. The black dashed diagonal represents the optimal calibration when the accuracy always equals the confidence. We can observe that for both CNN and BERT, tag representations learned with knowledge distillation consistently bring model calibration closer to the optimal line. After temperature-scaling, calibration of all compared models are further optimized.

4.2.4 Research Question 4

We explore the robustness of tag representations learned with knowledge distillation model against two types of adversarial attacks (Hsieh et al., 2019). The goal of a success attack is to fool the model to give the false privacy prediction by replacing one tag in the original input image tags. In this work we consider two types of attacks.
(a) Calibration without temperature scaling.
(b) Calibration results with temperature scaling.

Figure 3: Calibration results of TagCNN, TagCNN$_{KD}$, BERT, and BERT$_{KD}$. Dashed line is the optimal calibration function.

<table>
<thead>
<tr>
<th>Model</th>
<th>Trust</th>
<th>Surprise</th>
<th>Sadness</th>
<th>Joy</th>
<th>Fear</th>
<th>Disgust</th>
<th>Anticipation</th>
<th>Anger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private</td>
<td>32.33%</td>
<td>4.72%</td>
<td>10.33%</td>
<td>25.06%</td>
<td>11.00%</td>
<td>8.99%</td>
<td>5.91%</td>
<td>1.65%</td>
</tr>
<tr>
<td>Public</td>
<td>29.42%</td>
<td>6.05%</td>
<td>9.95%</td>
<td>22.96%</td>
<td>17.00%</td>
<td>5.87%</td>
<td>6.35%</td>
<td>2.40%</td>
</tr>
</tbody>
</table>

Table 4: Emotion distributions of tags attached to private and public images.

**Random Attack.** This type of attack randomly select one image tag and replace it with another word that is randomly selected from the vocabulary of the dataset.

**Synonym-based Attack** Randomly selecting the word to replace from the vocabulary may change the meaning too much (e.g. replace "good" with "bad") which is not considered as good attacks (Hsieh et al., 2019; Niu et al., 2020). We explore the synonym-based attack: image tags are replaced by one of their synonyms. Particularly, for each image, we start by replacing the first tag with its synonyms. If none of the attacks successfully fool the model, we move to the next tag with the previous tag unchanged. This process is repeated until either the attack succeeds or all tags have been exhausted.

Experimental results addressing research question 3 are shown in Table 3. We evaluate the robustness of TagCNN, BERT, TagCNN$_{KD}$, and BERT$_{KD}$ that have been well-trained for the supervised learning task in Section 4.2.1 against adversarial attacks. As suggested in (Hsieh et al., 2019), we randomly pick 100 examples from the test set that all models correctly predict, based on which we generate adversarial attacks. For random attacks, we repeat the process by $10^3$ times and calculate the average as the final performance. For the synonym-based attack, all synonyms are selected from WordNet. From Table 3 we observe that tag representations with knowledge distillation approach improve the robustness of BERT and TagCNN against the two types of adversarial attacks, especially for the private class. Moreover, we also notice that TagCNN$_{KD}$ does not show stronger robustness than its teacher model BERT, whereas BERT$_{KD}$ outperforms BERT. This result further suggests the advantage of self-distillation on BERT.

### 4.2.5 Research Question 5

We perform analysis to study the correlation between the privacy (private or public) and the emotion of image tags. Precisely, we observe the distributions of tags with various emotions for both the private and the public class to understand whether tags with certain emotions are more often used in private or public images and the underlying reasons behind it. We use the NRC Emotion lexicon (Mohammad and Turney, 2013), a lexicon of 10,000 words, each is associated with one of the eight emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive). In our experiment, we first randomly select the same number of private and public images (7000 for each class), and find common tags that exists in both image tags and NRC lexicon. The distribution of tags with eight emotions in both the private and the public class is shown in Table
Table 5: Examples of private and public images and corresponding tags associated with various emotions. Tags with specific emotions are colored in red.

<table>
<thead>
<tr>
<th>Emotion</th>
<th>Privacy</th>
<th>Trust private</th>
<th>Disgust private</th>
<th>Surprise public</th>
<th>Fear public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Positive</td>
<td>60.71%</td>
<td>39.29%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Private</td>
<td>60.71%</td>
<td>39.29%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public</td>
<td>55.71%</td>
<td>44.29%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Positive and negative emotion distributions of tags in private and public images.

4, where we observe that tags with emotions of trust, joy, and disgust are more often used to depict private images, while tags with fear and surprise emotions are more often attached to public images. Emotions of sadness, anticipation, and anger do not show obvious bias towards either privacy class.

Next, we look into some examples to better understand the underlying reasons behind such correlations. Examples of online images with tags of various emotions are shown in Table 5. We observe that the tag "innocent" with the emotion of trust is often used to depict images about children, which in many cases are considered as private images. Tags such as "messy" with the disgust emotion is often attached to images with indoor environments, which are ones that more often bias towards the private side. In contrast, tags such as "marvel" and "caution" with emotions of surprise and fear, respectively, are more often used to describe public constructions or signs, and thus are more often used in public images.

Moreover, we also analyze the distribution of tags with positive and negative sentiments for the private and the public class. Results are shown in Table 6, where we observe that tags with positive sentiments takes higher percentage in private images over public images.

5 Conclusion

In this work, we explore learning tag representations with knowledge distillation approach based on BERT for tag-based image privacy prediction. Our approach significantly outperforms the state-of-art models for tag-based image privacy prediction. We also perform a BERT-based semi-supervised learning approach using only a small amount of annotated data, where BERT achieves comparable performance with its supervise counterpart with only 20% of labeled data provided. Moreover, we also perform calibration analysis and show that tag representations learned with knowledge distillation yield better calibration. We also study the robustness of our learned tag representations against some adversarial attacks for image tags. Our results show that our approach show stronger robustness over compared baselines against random and synonym-based attacks. Last, we analyze the correlation between the privacy and the emotion of image tags and use some examples in the PicAlert dataset (Zerr et al., 2012a) to help us understand the underlying reasons.

Our future direction is to integrate deep CNN models for image processing with BERT to develop a multi-modal image privacy prediction model with both images and tags as inputs.

Acknowledgments

This research is supported in part by NSF. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of NSF. The computing for this project was performed on AWS. We also thank our reviewers for their feedback.

References


1627


Delexicalized Cross-lingual Dependency Parsing for Xibe

He Zhou and Sandra Kübler
Indiana University
{hzh1, skuebler}@iu.edu

Abstract

Manually annotating a treebank is time-consuming and labor-intensive. We conduct delexicalized cross-lingual dependency parsing experiments, where we train the parser on one language and test on our target language. As our test case, we use Xibe, a severely under-resourced Tungusic language. We assume that choosing a closely related language as the source language will provide better results than more distant relatives. However, it is not clear how to determine those closely related languages. We investigate three different methods: choosing the typologically closest language, using LangRank, and choosing the most similar language based on perplexity.

We train parsing models on the selected languages using UDify and test on different genres of Xibe data. The results show that languages selected based on typology and perplexity scores outperform those predicted by LangRank; Japanese is the optimal source language. In determining the source language, proximity to the target language is more important than large training sizes. Parsing is also influenced by genre differences, but they have little influence as long as the training data is at least as complex as the target.

1 Introduction

For a severely low-resource language, constructing a dependency treebank is labor-intensive and time-consuming, and annotators are difficult to find. Expanding a small treebank via monolingual dependency parsing leads to suboptimal results since we lack enough training data to train a reliable parser. This situation has led to an increasing interest in techniques for supporting low-resource languages by taking advantage of high-resource languages together with methods for cross-lingual transfer (Meechan-Maddon and Nivre, 2019). This is facilitated by the Universal Dependencies (UD) project, which has resulted in a treebank collection covering a wide range of language, with the goal of facilitating multilingual parser development (Nivre et al., 2020). The latest release (v2.7) covers 183 treebanks in 104 languages (Zeman, 2020). In our current work, we carry out preliminary single-source cross-lingual delexicalized dependency parsing experiments for the Xibe language. With this method, we train a parser on the treebank of one source language and parse the target language, with both treebanks delexicalized to abstract away from lexical differences between the two languages.

Choosing the source language is crucial for single-source cross-lingual parsing. The optimal source language needs to be syntactically close to the target language as well as high-resourced. However, it is not obvious how to select this language. We investigate three methods for selecting the source language: We compare LangRank (Lin et al., 2019) and typology, and we investigate whether using perplexity as a similarity metric can approximate typological knowledge. Then we investigate whether the size of the source treebank or a genre mismatch affect the quality of the parser.

The remainder of this paper is organized as follows: Section 2 provides a short overview of Xibe syntax. In Section 3, we describe our research questions in more detail. In Section 4, we briefly summarize methods of cross-lingual transfer. The experimental settings are introduced in Section 5. We then explain the methods for selecting source languages in Section 6, and in Section 7, we discuss our results. We conclude in Section 8.

2 The Xibe Language and Treebank

Xibe is a Tungusic language. There are twelve languages in the Tungusic language family spoken in
Central and Eastern Asia, but the numbers of Tungusic speakers have never been large (Robbeets and Savelyev, 2020). The language of Xibe is the one with a comparatively larger amount of active speakers in the whole language family.

Xibe shares morphological and syntactic features with other transeurasian languages. The transeurasian languages have a very rich system of case marking through the use of affixes (or particles in Japanese and Korean). All transeurasian languages are head final, they use verb-final word order, and attributes, complements, and adjuncts precede their headwords. In Xibe, clausal constituents have a rigid Subject-Object-Verb (SOV) word order, and all phrasal categories are consistently head-final. Like other Tungusic languages, Xibe has agglutinative morphology, which mainly focuses on verbs in that verbs are marked for tense, aspect, mood and voice, as well as convers and participles.

Zhou et al. (2020) describe a Xibe treebank annotated in the Universal Dependencies framework, containing 810 trees. Figure 1 shows an example Xibe dependency tree. The matrix predicate is at the sentence final position with the object and oblique constituent, marked by corresponding case markers, preceding the verb and the present tense suffix -mbi attached to the verb root.

### 3 Research Questions

Our research focuses on cross-lingual dependency parsing using a single target language and concentrates on determining the best method to select the optimal source language. More specifically, we investigate the following questions.

**Question 1** What are the most important factors to consider when choosing the best source language(s)? We investigate three methods: one uses typological knowledge, the second uses LangRank (Lin et al., 2019), a machine learning approach to rank languages based on their relatedness. The third method uses POS n-gram perplexity to determine similarity between languages. Here, our goal is to determine whether perplexity can be used to model typological knowledge.

**Question 2** Ideally, an optimal source language should be closely related to the target language as well as high-resourced, since we need a sizable treebank that can be used for training the parser. However, is a large treebank size more important than syntactic similarity with the target language in delexicalized dependency parsing?

**Question 3** The Xibe treebank includes sentences from two different genres, grammar examples and news whereas most UD treebanks contain multiple written genres. Considering that the performance of the parsing models trained on one domain degrades on sentences drawn from a different domain (McClosky et al., 2010), we assume that this happens in our setting as well. Therefore, we investigate how a parser trained on multiple genres performs on the two target language genres. Is the mixture of genres in the source data robust enough to cover both of our target genres?

### 4 Related Work

Cross-lingual transfer learning has been useful in improving the accuracy of a low-resource target language and has been applied in a multitude of tasks (Lin et al., 2019). The process of cross-lingual transfer learning refers to resources and models from high-resource source languages to low-resource target languages on different levels.

There are four main cross-lingual parsing approaches for dependency parsing: annotation projection (Yarowsky et al., 2001; Hwa et al., 2005), model transfer (Zeman and Resnik, 2008; McDonald et al., 2011), treebank translation (Tiedemann et al., 2014; Tiedemann and Agić, 2016), and multilingual parsing models (Duong et al., 2015b; Ammar et al., 2016; Kondratyuk and Straka, 2019). The annotation projection approach requires parallel treebanks of both source language and target language, and the treebank translation approach requires a machine translation system, while in the
model transfer approach, models trained on source language treebanks are directly applied to parse target languages. A multilingual parsing model is either a model trained on one source language which is refined by taking advantage of similar structures shared with the target language, or a multilingual model using multilingual word clusters and embeddings or language-specific features.

The main challenge for cross-lingual parsing is to reduce the language discrepancies on different levels between the source language and the target language. To reduce the great differences in writing systems and vocabulary, Zeman and Resnik (2008) used delexicalization, based on the hypothesis that the interaction between morphology and syntax in two languages is similar; they applied this approach in parsing Swedish using Danish as the source language. Since this method does not require bilingual parallel data, it is extensively implemented combining with other features. McDonald et al. (2011) implemented the idea of delexicalizing the parsing models and adapting the parsers with a constraint driven learning algorithm that achieved accuracy gains. Søgaard (2011) improved the approach by Zeman and Resnik (2008) by selecting the source sentences that are most similar to the target language. Rosa and Žabokrtský (2015) trained an MSTParser model interpolation as an alternative for multi-source cross-lingual delexicalized dependency parser transfer. The work by Rosa (2015) involved the training of several independent parsers which were applied to the same input sentence. The resulting tree was obtained by finding the maximum spanning tree of a weighted directed graph of the potential parse tree edges from the different parsers.

In addition to delexicalized methods, cross-lingual lexical representations can also be used in dependency parsing. Täckström et al. (2012) used parallel data to induce cross-lingual word clusters, and added them as features for their delexicalized parser. Xiao and Guo (2014) proposed that the source and target language words with the same meaning share a common embedding. The embeddings are jointly trained with a neural model and are used for dependency parsing. Duong et al. (2015a); Ahmad et al. (2019); He et al. (2019) proposed different methods to develop multilingual word representations and used them for dependency parsing. Also, these approaches utilize zero-shot parsing since the trained parsing models parse a target language without any training target instances (Romera-Paredes and Torr, 2015). It is a suitable method for parsing low-resource languages because knowledge between different languages is transferable and labeled low-resource language data is difficult to obtain.

For Xibe, there are currently no parallel corpora or machine translation systems available, which makes model transfer the most feasible approach. In order to achieve zero-shot single-source cross-lingual parsing, we first train a parsing model on one source language treebank, then parse the target language using this model. As Xibe is written in the traditional Mongolic alphabet, which differs greatly from all the candidate source languages, we must minimize these differences. Therefore, we use treebank delexicalization by replacing lexical items with only part-of-speech tags in both the source and target languages.

5 Experimental Settings

5.1 LangRank and Perplexity Calculation

5.1.1 LangRank

When predicting transfer languages, LangRank requires four types of input: a segmented target language dataset, an unsegmented target language dataset, target language code (in our case sjo) and task label (DEP). We use the 1 131 Xibe sentences (see Section 6.2) as the unsegmented dataset, and we create the segmented dataset with SentencePiece (Kudo and Richardson, 2018)\(^1\). SentencePiece is a language-independent sub-word tokenizer and detokenizer, which creates sub-word models directly from raw sentences, along with tokenization. Such a subword model is required by LangRank. We use the following SentencePiece parameters: We set the final vocabulary size to 8 000 since the Xibe dataset is small. We use the default value for the other two parameters, that is, the amount of characters covered by the model is set to 1.0, and the model type is set to unigram.

5.1.2 Perplexity

We compute perplexity scores based on POS bigrams. We build the bigram language models using NLTK (Bird, 2006) and use Laplace Smoothing to avoid zero probability for unseen bigrams, then calculate perplexity of each Xibe sentence over the

\(^1\)https://github.com/google/sentencepiece
source language model. The final score is averaged over all Xibe sentences per source language model.

5.2 Treebanks

The training data we use come from the Universal Dependencies (UD) project, version 2.7. That is, we retrieve treebanks of the candidate source languages described in Section 6. Since Turkish, Korean, and Japanese have multiple treebanks in UD, we use three Turkish treebanks: tr_gb, tr_imst and tr_boun, and three Japanese treebanks, ja_modern, ja_bccwj and ja_gsd. The perplexity score of ko_kais is 22.77 which is much higher than ko_gsd, we therefore only use ko_gsd (see Table 3). As for the remaining languages, if the language has more than one treebank, we only select the largest. We use the concatenation of train/dev/test splits per source language treebank as our training data. Moreover, the treebanks of candidate source languages differ from one another in size (see Table 4 and Table 5). bxx_bdt, kk_kdt, and ja_modern have only around 1 000 trees, the other treebanks range between around 3 000 and almost 90 000 trees. The size discrepancy is reduced by limiting each language to at most 3 000 trees, sampled randomly where necessary.

Since we use treebanks from the Universal Dependencies project, all treebanks share the same annotation scheme. However, we are aware that there may be differences in terms of annotation quality or the interpretation of language specific characteristics. Such issues are beyond the current project, but need to be addressed in future work.

The test data comes from the Xibe treebank, which generates three test datasets based on genre:
1. grammar: 544 grammar examples
2. news: 266 news sentences
3. mixed: the two genres combined, 810 trees

We delexicalize all the treebanks by replacing their word forms with their POS tags.

5.3 Parser

We use UDify (Kondratyuk and Straka, 2019) for the parsing experiments. UDify is a state-of-the-art multilingual multi-task model capable of accurately predicting universal parts of speech, morphology features, lemmas, and dependency trees simultaneously. It uses the pre-trained multilingual BERT model, which allows it to handle a large number of languages with reasonable performance, without requiring any language-specific components. On top of the BERT model, the parser uses an attention layer and a multi-task learning setup so that each of the linguistic tasks, predicting part-of-speech, morphological features, lemmas, and dependencies are single tasks that are learned jointly.

To determine whether UDify can parse Xibe straightforwardly without removing lexical items, we parse Xibe with the pre-trained UDify model, obtaining a UAS of 24.28% and an LAS of 6.79%. These results provide a strong indication that the vocabulary differences between Xibe and other languages cannot be bridged by the multilingual BERT model. Consequently, we decided to delexicalize our data and use (gold) POS sequences instead.

We train the individual models on the delexicalized treebank of each source language and parse the Xibe texts (also delexicalized) using those models. We use the default parameters, but set the warmup_steps and start_step to 256.

5.4 Evaluation

Evaluation is performed using the Unlabeled Attachment Score (UAS) and Labeled Attachment Score (LAS), as computed by the official evaluation script provided for the CoNLL 2018 shared task.3

6 Source Language Selection Methods

In this section, we describe the three methods for determining the best source languages, and present the languages chosen by these methods.

6.1 Typology Based Selection

The first approach uses linguistic knowledge: As described in Section 2, Xibe is a Tungusic language that shares morphological and syntactic features with other transuralian languages. Therefore, transuralian languages are assumed to be good candidates, including those belonging to Turkic, Mongolic, Tungusic, Koreanic and Japonic language families. To ensure that the candidate source languages have at least one dependency treebank, we limit our experiments to the following languages, which are included in the most re-

---

3https://universaldependencies.org/conll18/evaluation.html
Table 1: LangRank predictions on three genres.

<table>
<thead>
<tr>
<th>Genre</th>
<th>Top 3 predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>grammar examples</td>
<td>Czech ces</td>
</tr>
<tr>
<td></td>
<td>Norwegian nor</td>
</tr>
<tr>
<td></td>
<td>Spanish spa</td>
</tr>
<tr>
<td>news</td>
<td>Finnish fin</td>
</tr>
<tr>
<td></td>
<td>Slovenian slv</td>
</tr>
<tr>
<td></td>
<td>Korean kor</td>
</tr>
<tr>
<td>mixed</td>
<td>Finnish fin</td>
</tr>
<tr>
<td></td>
<td>Slovenian slv</td>
</tr>
<tr>
<td></td>
<td>Slovak slk</td>
</tr>
</tbody>
</table>

Table 2: Top 3 predictions using a single feature in LangRank.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Top 3 languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>geographic</td>
<td>Russian rus</td>
</tr>
<tr>
<td></td>
<td>Hindi hin</td>
</tr>
<tr>
<td></td>
<td>Latvian lav</td>
</tr>
<tr>
<td>genetic</td>
<td>Latvian lav</td>
</tr>
<tr>
<td></td>
<td>Czech ces</td>
</tr>
<tr>
<td></td>
<td>Norwegian nor</td>
</tr>
<tr>
<td>word overlap</td>
<td>Chinese zho</td>
</tr>
<tr>
<td></td>
<td>Indonesian ind</td>
</tr>
<tr>
<td></td>
<td>English eng</td>
</tr>
</tbody>
</table>

6.2 LangRank Based Selection

LangRank (Lin et al., 2019) is an approach for choosing source languages for cross-lingual NLP tasks including machine translation, entity linking, part-of-speech tagging, and dependency parsing. The task of selecting the optimal source languages for an NLP task is formulated as a ranking problem. Given a low-resource target language and a set of candidate source languages, a model is trained to rank the source languages according to the performance achieved when they are used in training to process the target language. Each candidate source language is represented with a set of dataset-dependent and dataset-independent features. The dataset-dependent features include dataset size, type-token ratio, word overlap and subword overlap, and the dataset-independent features include geographic distance, genetic distance, inventory distance, syntactic distance, phonological distance, and feature distance. Based on these features, the system implements gradient boosted decision trees (GBDT; Ke et al. (2017)) to select the best transfer languages for the four NLP tasks.

In our experiment, we use the Xibe treebank sentences (544 grammar examples and 266 news sentences) for prediction. We collect 321 more sentences from news to keep the two genres balanced, since LangRank does not define how much data is needed. Note that we use sentences as input for LangRank, we do not delexicalize the data for this step.

Table 1 lists the top three predicted source languages for each genre. Czech, Norwegian and Spanish rank among the top three when we feed in grammar examples. Finnish, Slovenian and Korean are the top three predictions when only news is used as input. Additionally, Finnish and Slovenian are also top languages when mixed data is used, followed by Slovak.

Lin et al. (2019) mentioned the possibility that LangRank cannot generalize well on certain languages since it is trained only on a few languages for the particular tasks. To obtain more educated guesses for choosing the transfer language, they analyzed the learned models and extracted the most important features for given tasks. In the dependency parsing task, geographic distance, genetic distance and word overlap are features that yield good scores on their own. Table 2 lists the top 3 predictions when only one relevant feature is used. In Table 1 and Table 2, only Czech and Norwegian appear in both results. But the results can be explained more easily. For example, Russia and India are geographically closer to the area where Xibe is spoken, and Xibe has larger word overlap with Chinese as a result of long-term language contact.

6.3 Perplexity Based Selection

Here, we attempt to automatically approximate the typological approach by determining similarity via POS bigrams. We use perplexity as a similarity metric. Basically, we determine the optimal source languages among the languages covered by Universal Dependencies by computing the perplexity between each of the treebanks (see Section 5.2) and Xibe. As vocabularies and orthographies among languages differ greatly, we use POS bigrams instead of words to calculate perplexity. The inherent assumption is that the POS bigrams
Table 3: Perplexity scores for source languages.

<table>
<thead>
<tr>
<th>Language</th>
<th>ISO</th>
<th>Lang.family</th>
<th>Treebank</th>
<th>PP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buryat</td>
<td>bxr</td>
<td>Mongolic</td>
<td>bxr_bdt</td>
<td>7.93</td>
</tr>
<tr>
<td>Kazakh</td>
<td>kaz</td>
<td>Turkic</td>
<td>kk_ktb</td>
<td>8.30</td>
</tr>
<tr>
<td>Turkish</td>
<td>tur</td>
<td>Turkic</td>
<td>tr_imst</td>
<td>8.59</td>
</tr>
<tr>
<td>Uyghur</td>
<td>uig</td>
<td>Turkic</td>
<td>ug_udt</td>
<td>8.69</td>
</tr>
<tr>
<td>Turkish</td>
<td>tur</td>
<td>Turkic</td>
<td>tr_boun</td>
<td>9.22</td>
</tr>
<tr>
<td>Turkish</td>
<td>tur</td>
<td>Turkic</td>
<td>tr_gb</td>
<td>9.25</td>
</tr>
<tr>
<td>Japanese</td>
<td>jpn</td>
<td>Japonic</td>
<td>ja_modern</td>
<td>12.48</td>
</tr>
<tr>
<td>Japanese</td>
<td>jpn</td>
<td>Japonic</td>
<td>ja_bccwj</td>
<td>13.94</td>
</tr>
<tr>
<td>Japanese</td>
<td>jpn</td>
<td>Japonic</td>
<td>ja_gsd</td>
<td>14.36</td>
</tr>
<tr>
<td>Korean</td>
<td>kor</td>
<td>Koreanic</td>
<td>ko_gsd</td>
<td>13.27</td>
</tr>
<tr>
<td>Korean</td>
<td>kor</td>
<td>Koreanic</td>
<td>ko_kaist</td>
<td>22.77</td>
</tr>
</tbody>
</table>

We find that the perplexity scores of the Mongolic and Turkic languages are closest to Xibe (lowest perplexity) among all languages (see Table 3). Comparing these languages with the ones chosen based on typology in Section 6.1, there are considerable overlaps, except for Japanese and Korean. We also examine the perplexity scores for Korean-Xibe and Japanese-Xibe: These scores are higher than those for Turkic and Mongolic languages but lower than most of the other languages.

7 Results and Analysis

In this section, we provide the results for our three research questions.

7.1 How to Choose the Source Language?

Table 4 shows the parsing results for source languages selected by typology. Among the transeurasian languages, Kazakh achieved the highest LAS of 58.69% on grammar examples while Japanese achieved the highest LAS of 38.59% when tested on news and the highest LAS of 44.91% when tested on mixed data. On all three test datasets, Korean had the lowest LAS, with 40.54% on grammar examples, 29.16% on news and 33.41% on mixed genres. Table 5 shows the results for source languages selected by LangRank. Korean scored the highest LAS whereas the lowest was achieved by Spanish with 15.11% on grammar examples, 8.45% on news and 10.94% on mixed genres.

Based on Table 4, we find the most suitable source language to be Japanese. Training on the ja_gsd treebank results in the highest LAS for news and mixed genres, but its LAS for grammar examples is 3.19% lower than when training on Kazakh. This proves that Kazakh is more accurate than Japanese at labeling dependency relations. In terms of news and mixed genres, the gap with Japanese is actually larger, which we will investigate in section 7.3. In addition, Uyghur also performs well, its LAS on mixed genre is only 1.21% lower.

When using perplexity on POS bigrams to choose the source language, we assume that a low perplexity corresponds to a good match. However, when we compare the complexity scores in Table 3 and the parsing results in Tables 4 and 5, the situation is more complex: The Japanese treebank ja_gsd performs best in parsing even though it has a high perplexity score. The Korean treebank ko_gsd has a slightly lower perplexity than the Japanese ja_bccwj, but the Japanese LAS is about 11 points higher than the Korean LAS (on mixed). Similarly, Kazakh, Uyghur, and the Turkish tr_imst have similar perplexities, but the Kazakh and Uyghur LAS are about 10 points higher than the Turkish LAS (on mixed), even though the Kazakh treebank is by far the smallest. This shows that bigram POS perplexity is not an ideal measure of syntactic similarity, even though it performs better than LangRank.

As described in Section 6.2, standard LangRank may not be able to provide the best predictions. Therefore, we also investigate single features that are important for dependency parsing (see Table 2). According to the geographic feature, Hindi has the highest LAS 39.93% (on mixed genre, see Table 6). Similar to Xibe, Hindi has a Subject-Object-Verb (SOV) word order. Hence, we assume that the good performance of Hindi is a result of its syntactic similarity to Xibe rather than its geographic proximity. The genetic feature alone is not a good indicator for source language selection as all three languages achieve LAS around or below 20% (see Table 6). Languages selected via the word-overlap feature have poor results as well. On the mixed genre, Chinese achieves an LAS of 21.03% while Indonesian and English achieve only 13.97% and 13.22% respectively (see Table 6). Since we only used POS tags, we ignore borrowed Chinese words in Xibe sentences, and the higher performance of Chinese shows that Xibe is syntactically closer to Chinese than to English and Indonesian. Nevertheless, the LAS of Chinese is much lower than that of
Table 4: Parsing results with typologically related languages as source languages, based on perplexity.

<table>
<thead>
<tr>
<th>Language</th>
<th>Treebank name</th>
<th>Treebank size</th>
<th>Training size</th>
<th>grammar</th>
<th>news</th>
<th>mixed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UAS</td>
<td>LAS</td>
<td>UAS</td>
</tr>
<tr>
<td>Buryat</td>
<td>bxr_bdt</td>
<td>927</td>
<td>927</td>
<td>65.00</td>
<td>43.70</td>
<td>44.89</td>
</tr>
<tr>
<td>Kazakh</td>
<td>kk_kdt</td>
<td>1078</td>
<td>1078</td>
<td>72.17</td>
<td>58.69</td>
<td></td>
</tr>
<tr>
<td>Turkish</td>
<td>tr_gb</td>
<td>2880</td>
<td>2880</td>
<td>69.22</td>
<td>50.39</td>
<td>33.08</td>
</tr>
<tr>
<td>Turkish</td>
<td>tr_inst</td>
<td>5635</td>
<td>3000</td>
<td>65.03</td>
<td>43.74</td>
<td>48.04</td>
</tr>
<tr>
<td>Turkish</td>
<td>tr_boun</td>
<td>9761</td>
<td>3000</td>
<td>66.61</td>
<td>47.21</td>
<td>51.97</td>
</tr>
<tr>
<td>Uyghur</td>
<td>ug_udt</td>
<td>3456</td>
<td>3000</td>
<td>69.45</td>
<td>52.48</td>
<td>54.60</td>
</tr>
<tr>
<td>Korean</td>
<td>ko_gsd</td>
<td>6339</td>
<td>3000</td>
<td>54.35</td>
<td>40.54</td>
<td>45.40</td>
</tr>
<tr>
<td>Japanese</td>
<td>ja_modern</td>
<td>822</td>
<td>822</td>
<td>69.94</td>
<td>52.42</td>
<td>51.82</td>
</tr>
<tr>
<td>Japanese</td>
<td>ja_bccwj</td>
<td>57028</td>
<td>3000</td>
<td>73.34</td>
<td>55.41</td>
<td>53.68</td>
</tr>
<tr>
<td>Japanese</td>
<td>ja_gsd</td>
<td>8071</td>
<td>3000</td>
<td>73.68</td>
<td>55.50</td>
<td>55.01</td>
</tr>
</tbody>
</table>

Table 5: Parsing results for languages chosen by LangRank.

<table>
<thead>
<tr>
<th>Language</th>
<th>Treebank name</th>
<th>Treebank size</th>
<th>Training size</th>
<th>grammar</th>
<th>news</th>
<th>mixed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UAS</td>
<td>LAS</td>
<td>UAS</td>
</tr>
<tr>
<td>Czech</td>
<td>cs_pdt</td>
<td>87913</td>
<td>3000</td>
<td>31.74</td>
<td>19.68</td>
<td>18.13</td>
</tr>
<tr>
<td>Norwegian</td>
<td>no_bokmaal</td>
<td>20044</td>
<td>3000</td>
<td>33.35</td>
<td>21.87</td>
<td>21.02</td>
</tr>
<tr>
<td>Spanish</td>
<td>es_ancona</td>
<td>17680</td>
<td>3000</td>
<td>23.03</td>
<td>15.11</td>
<td>13.87</td>
</tr>
<tr>
<td>Finnish</td>
<td>fi_ftb</td>
<td>18723</td>
<td>3000</td>
<td>52.72</td>
<td>37.47</td>
<td>34.32</td>
</tr>
<tr>
<td>Slovenian</td>
<td>sl_ssj</td>
<td>8000</td>
<td>3000</td>
<td>32.64</td>
<td>19.66</td>
<td>18.68</td>
</tr>
<tr>
<td>Korean</td>
<td>ko_gsd</td>
<td>6339</td>
<td>3000</td>
<td>54.35</td>
<td>40.54</td>
<td>41.26</td>
</tr>
<tr>
<td>Slovak</td>
<td>sk_snk</td>
<td>10604</td>
<td>3000</td>
<td>30.19</td>
<td>19.94</td>
<td>14.48</td>
</tr>
</tbody>
</table>

7.2 Syntactic Similarity vs. Data Size

In the previous section, we have found Japanese to be the optimal source language for Xibe, followed by Uyghur and Kazakh. However, the Kazakh treebank only contains 1,078 trees while the Japanese ja_gsd and Uyghur models are trained with 3,000 trees. We investigate whether the training set size is the main factor in reaching good parsing accuracy. Consequently, we sample 1,000 trees from the Japanese ja_gsd treebank, making the training set size comparable to Kazakh kk_kdt. Parsing results are displayed in Table 7. On all three test datasets, when training with 1,000 trees, the LAS slightly decreases compared to 3,000 trees. Despite this, both LAS and UAS are still higher for the 1,000 Japanese trees than for Kazakh, with the exception of the LAS on the grammar examples. This shows clearly that the training set size is contributing only minimally.

As the Japanese results increase slightly when increasing training data from 1,000 to 3,000 trees, an obvious question is whether we can improve results by increasing the training set size further. Thus, we train parsing models by sampling 6,000 trees from ja_gsd and using all 8,071 trees respectively (see Table 7). However, we only see a minimal increase in LAS (45.03% vs. 44.91%) and a small decrease in UAS (on mixed). Thus we can conclude that larger training data do not necessarily lead to an improvement in performance.

We also had a closer look at Japanese and Korean, which share many linguistic features, despite which Japanese performs better than Korean. On mixed data, Korean obtains an LAS of 33.41%. One possible reason for such a large gap can be found in the differences in annotations between the two languages. As described by Han et al. (2020), in the UD Korean treebanks, a sentence is segmented into eojeols. An eojeol can consist of lexical morphemes and functional morphemes, which means the functional morpheme is agglutinated to the lexical item preceding it. In contrast, the Japanese treebank adopts the Short Unit Word (SUW). This means that functional morphemes are annotated as separate units in the Japanese treebank, and their dependency relations are present. In Xibe, function words are written as any transeurasian language in Table 4, even lower than Korean by 12.38 points.
Table 6: Parsing results for source languages chosen using a single feature in LangRank.

<table>
<thead>
<tr>
<th>Language</th>
<th>Treebank name</th>
<th>Treebank size</th>
<th>Training size</th>
<th>grammar UAS</th>
<th>grammar LAS</th>
<th>news UAS</th>
<th>news LAS</th>
<th>mixed UAS</th>
<th>mixed LAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>geographic feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Russian</td>
<td>ru_syntagrus</td>
<td>61 889</td>
<td>3 000</td>
<td>28.14</td>
<td>17.84</td>
<td>18.79</td>
<td>9.42</td>
<td>22.28</td>
<td>12.56</td>
</tr>
<tr>
<td>Hindi</td>
<td>hi_hdtb</td>
<td>16 647</td>
<td>3 000</td>
<td>67.99</td>
<td>50.65</td>
<td>50.80</td>
<td>33.53</td>
<td>57.22</td>
<td>39.93</td>
</tr>
<tr>
<td>Latvian</td>
<td>lv_lvtb</td>
<td>13 643</td>
<td>3 000</td>
<td>37.78</td>
<td>23.68</td>
<td>26.67</td>
<td>18.36</td>
<td>30.82</td>
<td>20.35</td>
</tr>
<tr>
<td>genetic feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latvian</td>
<td>lv_lvtb</td>
<td>13 643</td>
<td>3 000</td>
<td>37.78</td>
<td>23.68</td>
<td>26.67</td>
<td>18.36</td>
<td>30.82</td>
<td>20.35</td>
</tr>
<tr>
<td>Czech</td>
<td>cs_pdt</td>
<td>87 913</td>
<td>3 000</td>
<td>31.74</td>
<td>19.68</td>
<td>18.13</td>
<td>10.54</td>
<td>23.00</td>
<td>13.95</td>
</tr>
<tr>
<td>Norwegian</td>
<td>no_bokmaal</td>
<td>20 044</td>
<td>3 000</td>
<td>33.35</td>
<td>21.87</td>
<td>21.02</td>
<td>14.48</td>
<td>25.63</td>
<td>17.25</td>
</tr>
<tr>
<td>word-overlap feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chinese</td>
<td>zh_gsdsimp</td>
<td>4 997</td>
<td>3 000</td>
<td>47.16</td>
<td>25.67</td>
<td>33.80</td>
<td>18.26</td>
<td>38.80</td>
<td>21.03</td>
</tr>
<tr>
<td>Indonesian</td>
<td>in_gsd</td>
<td>5 593</td>
<td>3 000</td>
<td>23.40</td>
<td>16.28</td>
<td>16.39</td>
<td>12.60</td>
<td>19.01</td>
<td>13.97</td>
</tr>
<tr>
<td>English</td>
<td>en_ewt</td>
<td>16 662</td>
<td>3 000</td>
<td>28.51</td>
<td>18.45</td>
<td>19.03</td>
<td>10.10</td>
<td>22.58</td>
<td>13.22</td>
</tr>
</tbody>
</table>

Table 7: Parsing results with sampling different amounts of data from ja_gsd

<table>
<thead>
<tr>
<th>Language</th>
<th>Treebank name</th>
<th>Treebank size</th>
<th>Training size</th>
<th>grammar UAS</th>
<th>grammar LAS</th>
<th>news UAS</th>
<th>news LAS</th>
<th>mixed UAS</th>
<th>mixed LAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japanese</td>
<td>ja_gsd</td>
<td>8071</td>
<td>1000</td>
<td>72.57</td>
<td>54.79</td>
<td>53.27</td>
<td>37.74</td>
<td>60.48</td>
<td>44.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3000</td>
<td></td>
<td>73.68</td>
<td>55.50</td>
<td>55.01</td>
<td>38.59</td>
<td>61.99</td>
<td>44.91</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6000</td>
<td></td>
<td>73.16</td>
<td>55.62</td>
<td>54.43</td>
<td>38.38</td>
<td>61.43</td>
<td>44.82</td>
</tr>
<tr>
<td></td>
<td>8071</td>
<td></td>
<td></td>
<td>73.20</td>
<td>55.74</td>
<td>54.33</td>
<td>38.64</td>
<td>61.39</td>
<td>45.03</td>
</tr>
</tbody>
</table>

6.3 Does Genre Matter?

As shown in Section 5.2, the Xibe treebank consists of two different genres (grammar and news) while most of the source treebanks have multiple genres. This design allows us to see how different genres influence parsing results. One prominent difference between Xibe grammar examples and news is that news sentences are much longer and use more complex syntactic structure. Thus, we expect to reach higher accuracy on the grammar examples. This is born out by the results in Tables 4 and 5: Among transeurasian languages in Table 4, Turkish tr_gb has the largest LAS difference between grammar examples and news by 26.48% whereas Uyghur has the smallest by 9.44%. In Table 5, Finnish displays the largest LAS discrepancy between the two genres by 11.45% whereas Spanish has the smallest difference by 6.66%. We find a general tendency that results on grammar are considerably higher than those on news, with sizable differences.

We now have a closer look at the three Turkish treebanks since tr_gb mainly contains grammar examples while the other two contain news and non-fictional data. Comparing performance of the models trained on the three treebanks, when we test with grammar examples, tr_gb outperforms the other two even though it is smaller in size. When testing on news, tr_boun and tr_imst reach similar results: tr_boun reaches an LAS of 37.77% and tr_imst reaches an LAS of 32.77%. However, tr_gb declines by 13.86% in LAS compared to tr_boun. The results indicate that genre does influence parsing. When the training data contains mainly simpler syntactic structures than the test data, the parser cannot analyze the more complex test data adequately.

8 Conclusion

In this research, we have investigated cross-lingual dependency parsing for Xibe. As we do not have parallel data or a machine translation system for this language, we delexicalize treebanks to avoid orthographic and lexical differences. We propose three criteria to select source languages, that is, typology, perplexity, and automatic predictions by the LangRank tool. Then, we train parsing models with UDify and test them with the three sets of
Xibe data. Our results demonstrate that syntactic similarity is considered the most important factor in delexicalized cross-lingual parsing. Japanese is found to be the optimal source language for parsing Xibe. Differences in genre also influence parsing. Parsers trained on simpler sentence structures cannot analyze more complex test data.

In our current work, we use only one source language to parse Xibe. We will determine the best concatenation of source languages for multilingual parsing in the future. Additionally, we will resegment the current units of the Korean treebanks into smaller units and create dependency relations by rules in order to determine if a more similar segmentation will lead to an improvement. Alternatively, we can use lexical information in parsing, such as creating a bilingual dictionary or training Xibe word embeddings.

Acknowledgments

We would thank the three anonymous reviewers for their helpful comments. He Zhou is supported by China Scholarship Council.

References


AutoChart: A Dataset for Chart-to-Text Generation Task

Jiawen Zhu1, Jinye Ran3, Roy Ka-wei Lee1, Kenny Choo1, and Zhi Li2

1Singapore University of Technology and Design
2University of Saskatchewan
3China Merchants Bank

{jiawen_zhu,roy_lee,kenny_choo}@sutd.edu.sg
z.li@usask.ca,rjy777@163.com

Abstract

The analytical description of charts is an exciting and important research area with many applications in academia and industry. Yet, this challenging task has received limited attention from the computational linguistics research community. This paper proposes AutoChart, a large dataset for the analytical description of charts, which aims to encourage more research into this important area. Specifically, we offer a novel framework that generates the charts and their analytical description automatically. We conducted extensive human and machine evaluations on the generated charts and descriptions and demonstrate that the generated texts are informative, coherent, and relevant to the corresponding charts.

1 Introduction

Natural language generation (NLG) is one of the core research areas in artificial intelligence (Gatt and Krahmer, 2018). Recent NLG studies have explored data-to-text generation, where exciting applications such as automated news reporting (Leppänen et al., 2017) were developed to generate text from non-linguistic data automatically. In this paper, we explore the chart-to-text generation problem, where analytical textual descriptions are automatically generated for a given graphical chart.

Chart-to-text generation has many exciting academic and commercial applications. For instance, preliminary analyses can be generated on charts to aid users in authoring analytical documents. On the accessibility front, automatically generated chart analyses can also support accessibility since text descriptions can be fed into speech-to-text modules and help visually impaired individuals to understand charts. Chart-to-text generation could also be applied to aid academic writing. Text descriptions of visual elements such as diagrams, charts, and graphs, are among the core academic assignments in linguistics (Molle and Prior, 2008). For example, the IELTS Academic Writing Task 1 (AWT1) is an assessment task that elicits written responses on a visual-verbal relationship. The AWT1 requires test takers to “describe, summarise, or explain the information in a graph, table, chart, or diagram.” Figure 1 shows an example of the AWT1. Chart-to-text generation offers the potential to generate large-scale chart analytical description learning examples for students attempting AWT1.

Despite the many benefits and applications of chart-to-text generation, this NLG task has received limited attention from computational linguistics and NLG researchers. Among the key factors that hinder the development of this research area is the lack of a large chart description dataset that may facilitate chart description studies. Intuitively, one possible solution is to collect and manually anno-
tate a chart description. For instance, we will first 
need to obtain a large dataset of charts and subse-
quently engage human annotators to write the sum-
mary and explanations for these charts. However, 
such a data collection process is time-consuming 
and expensive. Another approach is to perform 
large-scale data-crawling to retrieve charts and cor-
responding human-written summaries from the In-
ternet. However, it is challenging to ensure that text 
summaries correctly describe the chart and have 
provided adequate details to aid readers in under-
standing the chart as the charts are retrieved from 
multiple sources. For instance, in a recent study, 
(Obeid, 2020) had performed a large-scale data 
collection of charts and corresponding text descrip-
tions. However, the descriptions of the chart in the 
dataset contained background knowledge beyond 
the data illustrated in the chart.

In this paper, we aim to address chart analysis 
data scarcity and quality problems by proposing 
a novel framework that generates charts and their corresponding high-quality descriptions automatically. The AutoChart\textsuperscript{1} dataset generated by our proposed framework will pioneer new computational linguistic and NLG research area on chart descriptions. For instance, the availability of a large-scale chart description dataset encourages the creation of supervised machine learning and NLP models to interpret the charts and generate relevant text descriptions automatically.

We summarize our contributions as follows:

- We propose a novel framework to generate charts and their corresponding analytical descriptions automatically.

- Using our novel framework, we constructed AutoChart, which is a large-scale chart description dataset, and make this openly available to encourage future research.

- We conducted extensive human and machine evaluation on the generated charts and descriptions and demonstrate that the generated text is informative, coherent, and relevant to the corresponding charts.

2 Related Work

There are very few data-to-text works that investi-
gate chart recognition and understanding. Many of these existing works focused on extracting data

\textsuperscript{1}Code: https://gitlab.com/bottle\_shop/snlg/chart/autochart

from the various types of visual charts using deep learning computer vision and object recognition techniques (Cliche et al., 2017; Balaji et al., 2018; Liu et al., 2019; Ma et al., 2018; Dai et al., 2018; Battle et al., 2018; Chai et al., 2020). For instance, Balaji et al. (2018) proposed an automated system that extracted data points from bar and pie charts to create textual descriptions. However, the generated textual descriptions listed data values extracted from the figures in a static format without any analytical discussion about the charts’ overall trends or summary. Another line of work have also proposed table-to-text models (Iso et al., 2019; Puduppully et al., 2019), which aims to generate long and good-quality description from structured data formatted in a table. Nevertheless, these table-to-text models are designed for specific domains and structured data, and it is challenging to adopt these methods in the chart-to-text task.

Another related sub-domain of work is the visual-based question and answer (Q&A) task. Kahou et al. (2017) introduced the FigureQA corpus, which consists of over one million question-answer pairs grounded in over 100,000 visual charts. Methani et al. (2020) extended the work in (Kahou et al., 2017) and proposed the PlotQA corpus, which is a larger dataset with 28.9 million question-answer pairs over 224,377 charts from real-world sources and questions based on crowdsourced question templates. While large datasets have been collected for the visual-based Q&A task, these datasets are not applicable to generate analytical chart descriptions as the question-answer pairs are often short and data-specific without any in-depth analysis on the charts.

Closer to our work, Obeid and Hoque (2020) introduced a new large-scale corpus on chart summarization and proposed a transformer-based chart-to-text model. However, the descriptions of the chart in the dataset contained background knowledge beyond the data illustrated in the chart. These "noises" from the beyond-chart-data information may affect the learning of text generation models. Another prominent data source, Statista, has high-quality charts, but corresponding summaries may not be descriptive of the chart.

Our study addresses the limitations of existing chart-to-text datasets. It extends the existing works on chart recognition and data extraction by offering a novel framework to generate charts and their corresponding analytical descriptions auto-
matically. To this end, we construct and contribute AutoChart, a large-scale chart analytical description dataset.

3 AutoChart Dataset Construction

The goal of this study is to construct a dataset of charts with their corresponding analytical descriptions automatically. To this end, we propose a novel framework to construct the AutoChart dataset and illustrate its construction in Figure 2. We begin by collecting statistical data from multiple sources over the web and create the trend generation strategy. The goal of the strategy is to ensure that the generated charts exhibit some form of temporal trends, which ultimately encourages writers to identify these trends analytically. The proposed framework contains two main generation modules: chart generation and analytical description generation.

The statistical data and trend generation strategy guide the automatic generation of charts and their meta-information in the chart generation module. Specifically, we generate four types of charts: scatter plots, line charts, vertical and horizontal bar charts.

In the analytical description generation module, linguistic researchers are first recruited to write the analytical descriptions for a few charts. The human-written descriptions are used as templates for the automatic generation of analytical descriptions. As it is labor-intensive to draft human-written descriptions templates, we expand the number of templates by leveraging open-source algorithms to paraphrase the human-written descriptions. Subsequently, we analyze the linguistic rhetorical moves of the human-written and paraphrased templates. The rhetorical move analysis enables us to categorize the rhetorical function types of sentences presented in the analytical description templates.

Finally, the template sentences annotated with rhetorical moves are strategically sampled and adapted to chart data to generate the analytical description for a given chart.

3.1 Statistical Data Collection

To generate the charts, we first collected statistical data from multiple sources on the web, such as the World Bank Open Data and Nutritional Analysis Data. We crawled data from these sources to extract different variables whose relations could then be plotted (for example, a country’s labor force over time, etc.). There are a total of 346 unique indicator variables (CO2 emission, GDP growth, total population, etc.) with 76 unique entities (cities, states, countries, etc.). The data ranges from 1950 to 2016, though not all indicator variables have data items for all years. The data contains positive integers, floating-point values, and percentages. These values range from 0 to $3.50 \times 10^{15}$.

3.2 Trend Generation Strategy

Besides plotting the actual collected statistical data, we also aim to generate charts with specific trends. This encourages writers or machine learning algorithms to generate descriptions that analyze the patterns observed in the charts. To this end, we formulate a trend generation strategy, where data perturbation is applied to generate various types of trends. Specifically, we applied the following data perturbation:

$$Y = S_0e^{(\mu - \frac{\sigma^2}{2})x + \sigma W}$$  \hspace{1cm} (1)

Here $W$ denotes Brownian motion (Karatzas I.,
1998) that allows some degree of randomness in the trend generation, $S_0$ denotes the given initial value, $\sigma$ denotes the weight of Brownian motion, that is, the volatility rate of the data. $\mu - \frac{\sigma^2}{2}$ is the drift factor of Brownian motion, which indicates the trend of the data. When it is a positive number, the data is on an increasing trend, and when it is a negative number, it is on a decreasing trend. However, a random fluctuation is generated when it is 0. In total, we apply Equation 1 to generate charts with eight different types of trends. This is achieved by incorporating various parameters mentioned above and performing symmetry and rotation operations on the data. Figure 3 shows an example of line charts generated in various trends.

3.3 Chart Generation

We generate four types of charts in our AutoChart dataset: scatter plots, line chart, vertical, and horizontal bar charts. These types of charts are commonly encountered in academic journals, research papers, textbooks, etc.

Python library Matplotlib (Hunter, 2007) is used to generate the charts. To encourage diversity in our chart generation, we developed a script to select parameters randomly to add variation to our charts. Specifically, we randomly select markers from 10 unique shapes for each scatter-plot. We also randomly choose the color of the markers in scatter-plots, lines in line charts, and bars in bar charts from a set of 20 colors. The thickness of the bars and line style of lines are also randomly configured. Note that although we fix the size of the entire visual canvas, the size of legends and y-axis values is different for each chart, resulting in random image sizes. The number of discrete elements of x-axis varies from 2 to 8 and the number of entries in legend box varies from 1 to 2. By using different combinations of indicator variables, entities (years, countries, etc.), and parameters, we created a total of 10,232 charts.

Our script preserves the meta-information of the generated charts in JSON files to enable the development of supervised modules for various sub-tasks. Specifically, the meta-information contains bounding box annotations for the legend boxes, legend names and markers, axes labels, axes ticks, data coordinates, plot title, and image index. The meta-information will be used in the analytical description generation module to generate the charts’ corresponding descriptions. Furthermore, the meta-data could also be used in evaluating the correctness of future chart understanding models.

3.4 Analytical Description Generation

The creation of analytical descriptions for the generated charts is a challenging task. Firstly, as we have created a large number of charts, it is labor-intensive and time-consuming to draft the analytical descriptions for all the charts manually. Therefore, we would need an automated approach to generate the charts’ analytical description. Secondly, the automated solution would need to generate analytical descriptions that are informative, coherent, and relevant to chart context. We propose a template-based approach with linguistics analysis to guide the generation of charts’ analytical descriptions to overcome these challenges.

3.4.1 Templates Generation

We recruited three linguistics researchers to write the descriptions of a small subset of the generated charts to create the analytical description templates. The subset of generated charts is evenly sampled from the various types of trends. The linguistics researchers are instructed to assume the same setting as IELTS AWT1 when writing the analytical descriptions of sampled charts. In total, the linguistics researchers wrote analytical descriptions for 150 charts.

As writing the analytical descriptions templates is a labor-intensive and time-consuming task, we used Quillbot, an online paraphrase API, to paraphrase the sentences in the human-written templates. The paraphrase sentences significantly expanded our analytical description templates. In total, we extracted 213 human-written chart sentences, 661 paraphrased sentences as templates. Finally, both human-written and paraphrased sentences will be used to generate other generated charts’ analytical descriptions automatically.

\footnote{https://quillbot.com/}
3.4.2 Rhetorical Move Analysis

A naive and straightforward way to generate the charts’ analytical descriptions is to randomly sample the sentences from our templates and apply the charts’ meta-data to produce the relevant analytical descriptions. However, such an approach neglects the rhetorical moves in analytical descriptions, which are important linguistic elements in building analytical arguments (Swales, 2004). Inspired by the idea of moves from Swales’ framework of genre analysis, we explored a rhetorical moves framework in analytical description templates. Specifically, we manually annotate each sentence in the template and group them in one of the following five rhetorical moves:

1. **Move 1** [Obligatory]: Overview of the chart. This move is used to explain what the chart is about, the chart’s content, etc. For example, “The chart shows the amount of fast-food consumed in the UK between 1970 and 1990.”.

2. **Move 2** [Optional]: Description of the chart itself. This move mainly focuses on the configuration of or elements in the chart. For example, “All the sampled countries are from Europe: Finland, France, Georgia, Germany, Greece, and Hungary.”.

3. **Move 3** [Obligatory]: Interpretation of the chart information. This part mainly explains the changing trend and simple observation of chart information, etc. For example, “The amount of fish and chips eaten declined slightly”. Nevertheless, it is inadequate to simply describe the trends. Thus, we will add a supplementary **Move 3.1** to report the numeric information from the chart. For example, “In 1970, the consumption was about 300g per week. This fell to 220g per week in 1990.”. We noted that **Move 3.1** could be further divided into descriptions of individual data points and comparisons for trends.

4. **Move 4** [Optional]: Evaluative comments on specific value(s) or comparisons. For example, “The retired and unemployed people enjoyed about 78 to 82 hours per week which is longer than people from other employment statuses.”.

5. **Move 5** [Obligatory]: Conclusions, summaries or implications based on the chart. For example, “In conclusion, although there was a big increase in the consumption of pizza, sales of fish and chips decreased.”.

In particular, for sentences annotated as Move 3 or 4, we further categorize the sentences into the types of charts that they are describing:

- For temporal charts where the x-axis represents time, the sentences focus on the trend of the data and the comparison of different time points. Move 3 and 4 sentences that describe trends are grouped into the eight categories showed in Figure 3. For temporal charts without apparent trends, the sentences will mainly focus on the comparison between data and some special points.

- For categorical charts where the x-axis represents entities, such as cities, food, etc., the Move 3 and 4 sentences will only focus on comparing different categories and describing some special points.

3.4.3 Rhetorical Moves Synthesis for Chart Description

After analyzing and annotating the rhetorical moves of sentences in the human-written and paraphrase templates, we leverage the templates’ sentences and utilize charts’ meta-information to generate the charts’ analytical descriptions. To this end, we designed a script that takes in a generated chart as input and performs the following steps:

1. We first extract the generated chart’s data values and meta-information from its corresponding JSON file. Specifically, we extract the title, x-axis, and y-axis labels, numeric information, the data trend, etc.

2. Depending on the type of charts (i.e., temporal or categorical), and the trend(s) in the chart, we sample the sentences from the templates such that the sentences of various rhetorical moves are selected to build a coherent analytical description. Furthermore, to encourage diversity in the generated analytical description, we randomly set the number of rhetorical move sentences to generate. The conditional sampling of template sentences by rhetorical moves ensures that the generated analytical descriptions are structured to be a coherent analytical argument, and the sampling strategy encourages diversity in sentence structures.
3. Once the template sentences are selected, we replace the variables, entities, and values in the sentences with the given generated chart’s meta-information. For example, consider the template sentence “The [y-axis_label] of [x-axis_label] is observed to decline since [x-tick_label]”, we substitute the variables with the generated chart’s meta-information and generate the sentence “The number of visitors of Singapore is observed to decline since 2015.”. The script also analyzes corresponding relationships between data before performing the replacement if there is no related information in meta-data (i.e. the trend, statistical features such as minimum and maximum x-values and y-values, etc.). Such a process chooses templates randomly, and we can repeat the script three times to get multiple analytical chart descriptions for each chart.

Finally, the generated analytical descriptions are paired with the generated charts to form the AutoChart dataset.

4 Dataset Evaluation

In order to conduct a thorough evaluation on the generated analytical descriptions, similar to many NLG tasks, we assess the generated analytical descriptions using both human and automatic metrics (Gatt and Krahmer, 2018).

4.1 Dataset Overview

Table 1 summarizes our constructed AutoChart dataset. In total, we generated 10,232 charts and 23,543 corresponding analytical descriptions. Note that we have generated multiple analytical descriptions for each generated chart, simulating the real-world situation where different human writers may have different analytic descriptions of the same chart. The 150 analytical descriptions written by the linguistics researchers are also included in the dataset. The analytical descriptions have an average of 8 sentences and 140 words. Figure 4 shows an example of a generated chart and analytical description in the AutoChart dataset.

<table>
<thead>
<tr>
<th>Temporal</th>
<th>Trend</th>
<th>Horizontal</th>
<th>Vertical</th>
<th>Scatter</th>
<th>#Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temporal</td>
<td>Trend</td>
<td>880</td>
<td>480</td>
<td>880</td>
<td>880</td>
</tr>
<tr>
<td>Random</td>
<td>1,049</td>
<td>676</td>
<td>1,049</td>
<td>1,049</td>
<td>9,174</td>
</tr>
<tr>
<td>Categorical</td>
<td>951</td>
<td>436</td>
<td>951</td>
<td>951</td>
<td>7,564</td>
</tr>
<tr>
<td>Total</td>
<td>2,880</td>
<td>1,592</td>
<td>2,880</td>
<td>2,880</td>
<td>23,543</td>
</tr>
</tbody>
</table>

Table 1: Summary Statistics of AutoChart Dataset.

4.2 Human Evaluation

To examine the quality of the generated descriptions in AutoChart, we conducted three human-based evaluation studies. In the first study (S1), we recruited 30 linguistics researchers to write descriptions for 60 charts (20 line charts, 20 bar charts, and 20 scatter plots). The written descriptions from S1 are used in automatic evaluation discussed in the next section and also as the charts in S3. In Study 2 (S2) and Study 3 (S3), we examined the differences between AutoChart generated descriptions and the human-written descriptions from S1, respectively. They are the same otherwise in format and content. We studied S2 and S3 with 600 unique participants (20 line charts, 20 bar charts 20 scatter plots, each evaluated five times = 300 participants × 2 studies) using crowdsourcing on Amazon Mechanical Turk (AMT). Participants were at least 21 years old and were self-reported to be proficient in English. To reduce the potential bias in self-report, we used AMT’s options to select only US-based workers.

Informed consent was first obtained from participants. They then completed a demographics survey before proceeding to the study task. Participants were presented with a chart and its accompanying description, and then asked to rate the description on three dimensions of naturalness, informativeness, and quality (i.e., grammatical correctness) adapted from the study in (Novikova et al., 2018) using a 5-pt Likert scale. To ensure that participants were focused during the task, we asked them to answer a question that pertained to the chart description. We additionally used a reCAPTCHA (rec) to reduce the likelihood of bot responses. Five participants rate each chart, and we compute the median to provide majority voting in ratings.

Results. Comparing the results of S2 and S3, we did not detect significant differences between AutoChart and human-written descriptions for naturalness ($p = 0.056 > 0.05, 1$-tail), informativeness ($p = 0.288$) or quality ($p = 0.227$). From Figure 5, we observe that human descriptions are rated higher on dimensions of naturalness and marginally on quality; with the generated analytical descrip-
From 2013 to 2019, the line graph depicts the number of fast food (hamburger) consumption in Canada and the United States, respectively. In the last seven years, both countries have seen similar increases in consumer numbers. Over the last seven years, the United States has seen a steady increase. In 2018, there was a significant growth in Canada. Based on historical trends, both countries are anticipated to expand their fast food consumption in the coming years.

**Generated:** [Move 1] The line graph displays the number of consumption of fast food (hamburger) in Canada and the USA, respectively, from 2013 through 2019. [Move 2] In this chart, the unit of measurement is Local Currency, as seen on the y-axis. [Move 3] It is obvious that both countries shared similar increasing trends in the number of consumption in the past 6 years. [Move 3.1] For Canada, by 2013 the number of consumption reached nearly 12, while the number continued to increase until 34 in 2019. [Move 3.1] And for the USA, in 2013, the number of consumption was about 26, after that, each year has witnessed some increase. [Move 3] In the past 6 years, the USA had consistently more than Canada. [Move 5] It would be interesting to see what would happen in the next decade in these two countries in terms of current situations.

**Human:** From 2013 to 2019, the line graph depicts the number of fast food (hamburger) consumption in Canada and the United States, respectively. In the last seven years, both countries have seen similar increases in consumer numbers. Over the last seven years, the United States has seen a steady increase. In 2018, there was a significant growth in Canada. Based on historical trends, both countries are anticipated to expand their fast food consumption in the coming years.

**Figure 4:** Example of a generated chart and the corresponding human and automatic generated analytical descriptions in AutoChart dataset.

**Figure 5:** AutoChart vs Human descriptions rated on naturalness, informativeness, and quality

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU</th>
<th>ROUGE</th>
<th>BLEURT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoChart</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bar</td>
<td>40.21</td>
<td>42.99</td>
<td>21.42</td>
</tr>
<tr>
<td>Line</td>
<td>43.93</td>
<td>47.32</td>
<td>22.58</td>
</tr>
<tr>
<td>Scatter</td>
<td>39.69</td>
<td>48.03</td>
<td>17.30</td>
</tr>
<tr>
<td>Overall</td>
<td>41.28</td>
<td>46.11</td>
<td>20.43</td>
</tr>
<tr>
<td>Baseline</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bar</td>
<td>32.63</td>
<td>35.95</td>
<td>12.25</td>
</tr>
<tr>
<td>Line</td>
<td>35.48</td>
<td>33.20</td>
<td>7.55</td>
</tr>
<tr>
<td>Scatter</td>
<td>32.28</td>
<td>32.33</td>
<td>9.12</td>
</tr>
<tr>
<td>Overall</td>
<td>33.46</td>
<td>33.83</td>
<td>9.64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU</th>
<th>ROUGE</th>
<th>BLEURT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoChart</td>
<td>40.21</td>
<td>42.99</td>
<td>21.42</td>
</tr>
<tr>
<td>Baseline</td>
<td>32.63</td>
<td>35.95</td>
<td>12.25</td>
</tr>
</tbody>
</table>

**4.3 Automatic Evaluation**

Automatic evaluation of NLG tasks is challenging and an ongoing research area itself. The challenges of evaluating charts’ analytical description automatically are compounded as the generated text are significantly longer than other NLG task such as machine translation. Nevertheless, we leverage existing automatic evaluation metrics commonly used in NLG tasks to evaluate our generated text. Specifically, we perform two automatic assessments on the AutoChart dataset: (i) Quality assessment, which compares the automatic generated analytical descriptions and 60 human-written references written by the linguistics researchers in human evaluation study S1. (ii) Difficulty assessment, where to train existing chart-to-text methods using the AutoChart dataset and compare their generated descriptions against the human-written references.

**4.4 Quality Assessment**

To evaluate the quality of the analytical descriptions in AutoChart, we computed the ROUGE (Papineni et al., 2002), BLEU (Lin, 2004) and BLEURT (Selmam et al., 2020) scores between the human-written references from earlier human-based evaluation study S1 and the automatic generated analytical descriptions for the same 60 charts. We assume that the human-written references are the gold standard, and the generated analytical descriptions in AutoChart should be similar to the gold standard.
Method | BLEU | ROUGE | BLEURT
--- | --- | --- | ---
Balaji et al. (2018) | 20.45 | 22.9 | 13.31
Obeid (2020) | 33.05 | 28.32 | 18.23
Liu et al. (2020) | 10.68 | 19.74 | 5.49

Table 3: Difficulty Assessment Results.

As a baseline comparison, we adopt a simple template-based generative method that generates the charts' analytical descriptions by randomly sampling the sentences from our templates and applying the charts' meta-data to produce the relevant analytical description. The main difference between the baseline and the AutoChart analytical descriptions is the baseline does not consider the rhetorical moves in the description generation.

Table 2 shows the results of quality assessment on the analytical descriptions in AutoChart dataset and baseline. We compute the average scores for various automatic assessment metrics for the different chart types. The overall average scores are also reported. We observe that the AutoChart's analytical descriptions significantly outperformed the baseline generated text, suggesting that the inclusion of rhetorical moves in analytical descriptions are more aligned to the human-written references.

4.5 Difficulty Assessment

Besides evaluating the quality of the AutoChart dataset, we are also interested in investigating the existing chart-to-text methods’ performance in our new dataset. The goal is to assess the difficulty of generation chart analytical descriptions using the existing methods and the AutoChart dataset. Specifically, for this experiment, we first train the two state-of-the-art chart-to-text baselines (Balaji et al., 2018; Obeid, 2020) and an image captioning method (Liu et al., 2020) using the AutoChart dataset. Subsequently, we apply the trained baselines to generate the descriptions for the 60 charts in human evaluation study S1. Finally, we compute the ROUGE, BLEU, and BLEURT scores between the human-written references and the baselines’ generated descriptions of the charts.

Table 3 shows the experiment results. We observe that none of the methods can perform exceeding well in generating chart descriptions that are close to human references. The best performing baselines, (Obeid, 2020), was able to achieve similar results to the simple template-based generative baseline used in the quality assessment experiment. Unsurprisingly, the (Obeid, 2020) is not able to perform well for the chart analytical description generation task as the model did not consider the paragraph structure (i.e., rhetorical moves) in its generation. (Balaji et al., 2018) is designed to generate simple single sentence summaries for charts. Thus, it might not be able to generate informative and detailed analytical descriptions of the charts. The image caption method (Liu et al., 2020) performed badly for the task as it is likely to generate the general captions such as “this is a line chart.”.

The performance of existing baselines highlights the difficulty of the chart analytical description generation task.

5 Discussion and Conclusion

The AutoChart dataset opens up new research opportunities for the computer vision, computational linguistics, and natural language processing research communities. Novel object recognition and deep text generative models can be designed to interpret charts and generate relevant analytical descriptions automatically. The automatic interpretation and generation of analytical chart descriptions have many academic and industrial applications. For instance, generating good-quality analytic chart descriptions can guide students to attempt the IELTS AWT1. The automated analysis of charts is also a valuable function in existing assisted writing tools. The AutoChart dataset can support the development and exploration of the supervised chart-to-text methods.

We opined that this is the start of an emerging research topic, and many future works could be done. As an extension of this work, we aim to investigate and model more sophisticated linguistic techniques to construct better quality analytical descriptions of charts. We will expand the dataset to include more types of charts, e.g., pie charts, box plots, etc. Finally, we will also explore more automatic evaluation methods to assess the quality of the generated analytical descriptions. For example, we can examine and assess the analytical descriptions’ logic, reasoning, and fluency.

To conclude, we have proposed a novel framework that automatically constructs the AutoChart dataset, a large chart analytical description dataset. We conducted extensive human and machine evaluation on the generated charts and descriptions and demonstrate that the generated text is informative, coherent and relevant to the corresponding charts. We hope that the AutoChart can encourage more research in the automatic generation of analytical descriptions of charts.
Acknowledgement

This research is supported by Living Sky Technologies Ltd, Canada under its research exploratory funding initiatives. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of Living Sky Technologies Ltd, Canada.

References


Chengliang Chai, Guoliang Li, Ju Fan, and Yuyu Luo. 2020. Crowdchart: Crowdsourced data extraction from visualization charts. IEEE Transactions on Knowledge and Data Engineering.


A Comparative Study on Abstractive and Extractive Approaches In Summarization of European Legislation Documents

Valentin Zmiycharov  
FMI, Sofia University  
"St. Kliment Ohridski"  
Sofia, Bulgaria  
valentin.zmiycharov@gmail.com

Milen Chechev  
FMI, Sofia University  
"St. Kliment Ohridski"  
Sofia, Bulgaria  
milen.chechev@fmi.uni-sofia.bg

Gergana Lazarova  
FMI, Sofia University  
"St. Kliment Ohridski"  
Sofia, Bulgaria  
gerganal@fmi.uni-sofia.bg

Todor Tsonkov  
FMI, Sofia University  
"St. Kliment Ohridski"  
Sofia, Bulgaria  
ttsonkov@gmail.com

Ivan Koychev  
FMI, Sofia University  
"St. Kliment Ohridski"  
Sofia, Bulgaria  
koychev@fmi.uni-sofia.bg

Abstract

Extracting the most important part of legislation documents has great business value because the texts are usually very long and hard to understand. The aim of this article is to evaluate different algorithms for text summarization on EU legislation documents. The content contains domain-specific words. We collected a text summarization dataset of EU legal documents consisting of 1563 documents, in which the mean length of summaries is 424 words. Experiments were conducted with different algorithms using the new dataset. A simple extractive algorithm was selected as a baseline. Advanced extractive algorithms, which use encoders show better results than baseline. The best result measured by ROUGE scores was achieved by a fine-tuned abstractive T5 model, which was adapted to work with long texts.

1 Introduction

Automatic summarization of legislation documents is a rather challenging task, because they usually are very long and hard to understand. Therefore, any progress on this task might have great value for many businesses.

Most of the existing methods for text summarization are designed for a relatively short text (such as news, web pages, etc.). Most of the available datasets consist of short summaries of articles, news, etc. in which the expected summary is a few sentences long. However, in the case of legal documents both the text and the summary are longer. Dernoncourt et al. (2018) provide a comprehensive overview of the current datasets for summarization, including CNN/Daily Mail (Hermann et al., 2015), Gigaword (Graff and Cieri, 2003), LCSTS (Hu et al., 2015) and others. Noticeably, most of the larger scale summarization datasets consist of relatively short documents.

The paper’s aim is to research the applicability of different algorithms for text summarization on a new dataset, called EU legislation documents. The collected documents were parsed, cleaned and prepared to be used for data summarization training by defining pairs of full text and corresponding summary.

There are two main approaches for text summarization: extractive and abstractive. Extractive summarization means identifying important parts of the text and concatenating them verbatim to produce a summary which is a subset of the sentences from the original text. Abstractive summarization aims to make algorithms that are able to ”understand” the whole text and to generate a new shorter text that conveys the most important information from the original one (Sciforce, 2019).

The algorithm, which was selected as baseline, uses a classical approach for generating extractive summaries - sentence importance evaluation and combining the highly scored sentence to generate the summary (Malik, 2019). In this paper, we report results from different algorithms compared to the baseline algorithm. The compared algorithms include extractive and abstractive PreSumm (Liu and Lapata, 2019), which generates state-of-the-art results for CNN/DailyMail datasets (Hermann et al., 2015), a fine-tuned abstractive T5 model (Raffel et al., 2019), an extractive summarizer which uses BERT (Miller, 2019), an extractive summarizer which uses LEGAL-BERT (Chalkidis, 2015), Gigaword (Graff and Cieri, 2003), LCSTS (Hu et al., 2015) and others. Noticeably, most of the larger scale summarization datasets consist of relatively short documents.
et al., 2020).

2 Related Work

This section presents existing approaches for text summarization. Some are especially related to our work because they focus on long texts and this is the case with the legal documents that we use.

Xiao and Carenini (2019) focus on extracting informative sentences from a given document (without dealing with redundancy), especially when the document is relatively long (e.g., scientific articles). They rely on section information to guide the generation of summaries. Global and local contexts are taken into account when deciding if a sentence should be included in the summary. This approach struggles when there is not a well defined structure of sections which is the case with the legislation documents.

Nakao (2000) presents an algorithm for text summarization using the thematic hierarchy of a text. The proposed algorithm is intended to generate a one-page summary. Based on the ratio of source text size to a given summary size, the algorithm generates a summary with some breaks to indicate thematic changes. This algorithm cannot easily be adapted to summaries with dynamic length.

Another approach is to combine extractive and abstractive models (Wang et al., 2017). In the extraction phase, it creates a graph model to extract key sentences. In the abstraction phase, it uses a recurrent neural network based encoder-decoder, and devises pointer and attention mechanisms to generate summaries.

Vaswani et al. (2017) presented the Transformer architecture, which establishes a new single-model state-of-the-art BLEU score on two machine translation tasks. The architecture consisted of feed forward networks and attention mechanism. The basic architecture of a Transformer is based on the encoder-decoder model and is especially suitable for summarization because it can handle sequential data. Yet, the data does not need to be processed in order (for instance the beginning of the text does not have to be processed before the end). This is very useful for parallel training and reduces the time needed to train the transformers. The encoder takes all the input and encodes it into a vector containing the numerical representation of the text. Then the decoder decodes the vector and produces the summary. The datasets used for training can be big and thus exist pre-trained systems such as BERT (Bidirectional Encoder Representations from Transformers). They have been trained with huge general language datasets and can be fine-tuned to specific language tasks. The following algorithms we experimented with also rely on the Transformer architecture: PreSumm (Liu and Lapata, 2019), LEGAL-BERT (Chalkidis et al., 2020), T5 (Raffel et al., 2019).

Pegasus (Zhang et al., 2019) is a state-of-the-art NLP deep-learning algorithm for abstractive text summarization. It can be used for both extractive and abstractive summarization but the abstractive is more challenging because when the text is long, it should be understood, processed and a new text should be generated.

3 Dataset Collection

In order to create a dataset on which to compare the algorithms, we collected legislation documents. The data was preprocessed and only the relevant information for the task was left.

The dataset consists of short, easy to understand explanations of the main legal acts passed by the EU, intended for a general audience. Most cover the main types of legislation passed by the EU: directives, regulations and decisions. But some cover other documents, such as international agreements. The summaries are grouped into 32 policy fields, and each links to the full, official version of the act. Summaries are not available for legal acts that are considered to be already sufficiently short/clear or aimed exclusively at a specialist audience (Publications Office of the European Union, 2020). The information is provided by the Publications Office of the European Union and is publicly available. It was retrieved on January 12, 2020.

After the data was collected, we analyzed it and cleaned it in order to focus on the problem of text summarization. For the summaries we extracted only the Key points section and for the full documents we removed the references to external documents. There are some summaries that combine more than one full legislation document: 169 summaries are a summary of two documents and 50 summaries are a summary of more than two documents. In these cases, the full documents are concatenated in the mentioned order.

In order to be able to evaluate a wider variety of algorithms and remove potentially incorrect data, the outliers were handled in the following ways:

- 49 summaries with more words than full text
were removed from the dataset.

- 86 summaries, which summarize more than one document and the same document exists for more than one summary were removed from the dataset.

- 18 summaries with word count ratio bigger than 200 were removed. Words ratio has a mean value of 28 and value of 29 at third quantile.

- Nine full documents with more than 75000 words were removed. Full document word count has a mean value of 9615 and value of 11141 at third quantile.

- Three summaries with more than 1500 words were removed. Summaries’ word count has a mean value of 429 and value of 530 at third quantile.

- Two full documents with more than 2000 sentences were removed.

- Three full documents with sentence ratio more than 80 were removed.

- 29 summaries which summarize more than two documents were removed from the dataset.

During the initial collection the dataset contained 1750 records. After the cleaning there are 1563 summaries (10.7% of the dataset was removed). The mean length of the summaries and full texts is 424 and 8990 words respectively (see Fig. 1). The ratio between the number of sentences in summaries and full documents is 0.16.

4 Experiments

The aim of the experiments described in this section is to compare different approaches for text summarization of legislation documents. For this purpose we used the dataset mentioned in the previous section.

4.1 Experiments Design

Different algorithms were experimented on the same dataset. They contain both extractive and abstractive approaches. Some of them do not require training, while others are trained from scratch or fine-tuned on the dataset.

T5 and PreSumm (which is based on BERT) have restrictions for the number of tokens in the input and in the output. In order to be able to handle the data for these algorithms, the full texts and summaries were splitted into chunks: full texts containing 1024 tokens and summaries - 128 tokens. We used the same ROUGE metric to evaluate each summary chunk against each full text chunk. During training each full text chunk is paired with the most applicable summary chunk. During evaluation we generated summaries for all chunks from the full text. They were concatenated and the result...
was evaluated against the original summary.

4.1.1 Extractive Summarization Based on Weighted Frequency Tokens of Sentences (Baseline Algorithm)

The first approach that was used is basic extractive summarization (Malik, 2019). The first step of the algorithm is to split the full text into a list of sentences. After that all special characters and stop words are removed. Then all sentences are tokenized. Next the weighted frequency of occurrences of all words must be calculated. The weighted frequency of each word can be found by dividing its frequency by the frequency of the most occurring word. After that, the words in the original sentences are replaced by their respective weighted frequency. Weighted frequency for the words removed during preprocessing is zero. For each sentence, the sum of weighted frequencies is calculated. Only sentences with more than three words are evaluated in order to avoid the ones that do not contain enough information by themselves. Finally, the sentences are ordered in descending order by the sum of the weighted frequencies. The summary contains the sentences in the beginning of the ordered list. The number of sentences to be selected is based on the ratio between the number of sentences in the training dataset. The algorithm does not require training and is entirely based on the content of the full document.

4.1.2 Fine-tuned PreSumm Encoder

PreSumm is a pre-trained encoder based on BERT for the purpose of text summarization (Liu and Lapata, 2019). This method is entirely based on BERT and provides two implementations: BERT-SUMEXT for extractive summarization and BERT-SUMABS for abstractive summarization.

For both extractive and abstractive settings, the algorithm generates a summary consisting of the sentences which maximize the ROUGE-2 score against the gold summary during training. When generating summaries for a new document, the model is first used to obtain the score for each sentence. These sentences are ranked by their scores from highest to lowest. During sentence selection, Trigram Blocking is used to reduce redundancy (Paulus et al., 2017). Given a summary and candidate sentence, the sentence is skipped if there exists a trigram overlapping between it and the summary. The aim is to minimize the similarity between the sentence being considered and sentences which have been already selected as part of the summary.

4.1.3 Extractive Text Summarization with BERT and K-Means

We used the solution proposed by Miller (2019). It works the following way: the document is tokenized into clean sentences. The tokenized sentences are passed to the BERT model for inference to output embeddings. The embeddings are then clustered with K-Means. The embedded sentences that were closest to the centroid are selected as the candidate summary sentences. The algorithm uses the core BERT implementation. Fig. 2 displays a heatmap of the full text and the words that were selected to be part of the summary.
<table>
<thead>
<tr>
<th>Model</th>
<th>Type</th>
<th>Metric</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extractive summarization based extractive</td>
<td>Rouge 1</td>
<td>19.22</td>
<td>73.14</td>
<td>26.52</td>
<td></td>
</tr>
<tr>
<td>on weighted frequency tokens of sentences</td>
<td>Rouge 2</td>
<td>7.41</td>
<td>29.94</td>
<td>10.41</td>
<td></td>
</tr>
<tr>
<td>(baseline)</td>
<td>Rouge 3</td>
<td>3.46</td>
<td>13.30</td>
<td>4.83</td>
<td></td>
</tr>
<tr>
<td>Summarization with</td>
<td>Rouge 1</td>
<td>35.85</td>
<td>54.16</td>
<td>36.82</td>
<td></td>
</tr>
<tr>
<td>BERT and K-Means</td>
<td>Rouge 2</td>
<td>11.57</td>
<td>17.14</td>
<td>11.65</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rouge 3</td>
<td>4.64</td>
<td>6.23</td>
<td>4.48</td>
<td></td>
</tr>
<tr>
<td>Summarization with</td>
<td>Rouge 1</td>
<td>34.06</td>
<td>56.45</td>
<td>36.06</td>
<td></td>
</tr>
<tr>
<td>LEGAL-BERT and K-Means</td>
<td>Rouge 2</td>
<td>11.25</td>
<td>18.48</td>
<td>11.72</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rouge 3</td>
<td>4.63</td>
<td>6.95</td>
<td>4.63</td>
<td></td>
</tr>
<tr>
<td>PreSumm</td>
<td>Rouge 1</td>
<td>22.64</td>
<td>71.80</td>
<td>29.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rouge 2</td>
<td>8.19</td>
<td>28.20</td>
<td>10.85</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rouge 3</td>
<td>3.47</td>
<td>11.52</td>
<td>4.57</td>
<td></td>
</tr>
<tr>
<td>PreSumm</td>
<td>Rouge 1</td>
<td>33.30</td>
<td>25.09</td>
<td>28.46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rouge 2</td>
<td>5.41</td>
<td>4.08</td>
<td>4.63</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rouge 3</td>
<td>1.29</td>
<td>0.99</td>
<td>1.11</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>Rouge 1</td>
<td>42.89</td>
<td>52.25</td>
<td>39.27</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rouge 2</td>
<td>15.94</td>
<td>18.97</td>
<td>14.17</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rouge 3</td>
<td>7.28</td>
<td>8.07</td>
<td>6.28</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Results for experimented models. Fine-tuned T5 model generated the best results. Baseline F1 score was improved by all other algorithms.

4.1.4 Data-specific Approach for Legal Documents

The Extractive Text Summarization with BERT and K-Means allows its encoding model to be replaced and we experimented by changing it to LEGAL-BERT. (Chalkidis et al., 2020). LEGAL-BERT is a model which is based on BERT and is trained on twelve GB of diverse English legal texts from several fields. This experiment was encouraged by the specific vocabulary which the legislation documents consist of.

4.1.5 T5 Model

T5 (Raffel et al., 2019) is an encoder-decoder model and converts all NLP problems into a text-to-text format. It is trained using teacher forcing. This means that for training it always needs an input sequence and a target sequence. It is pre-trained on an open-source pre-training dataset, called the Colossal Clean Crawled Corpus (C4). The T5 model, pre-trained on C4, achieves state-of-the-art results on many NLP tasks while being flexible enough to be fine-tuned to a variety of important downstream tasks.

4.2 Evaluation Metrics

The most widely used metric for evaluation of text summarization is ROUGE (Recall-Oriented Understudy for Gisting Evaluation). ROUGE is a set of metrics used for evaluating automatic summarization and machine translation software. The metrics compare an automatically produced summary to a reference or a set of references (human produced) summary. ROUGE-N refers to the overlap of n-gram between the system and reference summaries. In particular ROUGE-1, ROUGE-2 and ROUGE-3 were used in the conducted experiments.

4.3 Results

Table 1 shows the results from the experiments. All extractive approaches outperform the baseline algorithm. PreSumm only improved Rouge 1 score from 26.52 (baseline) to 29.25. Extractive Text Summarization with BERT and K-Means yielded the best extractive results - 36.82 Rouge 1 score. When we tried to use the same algorithm but replaced BERT with pre-trained LEGAL-BERT which is fine-tuned on legal texts the results were slightly worse - 36.06 Rouge 1 score. Having in mind that the original summaries are generated in an abstractive way by experts the score of 36.82 can be considered a big success.

Both abstractive algorithms (PreSumm and T5) have limitations on input and output size. Both full texts and summaries were splitted to chunks of sizes 1024 and 128 respectively. During the
Fine-tuned T5-base abstractive model with overridden implementation for handling the long texts by splitting them to chunks showed the best overall results. Here is first paragraph the summary with the highest score (0.65 Rouge-1 F1 score):

Original:

The European order for payment (EOP) procedure applies to all civil and commercial matters in cases where at least one of the parties lives in an EU country different from the one where the application for an order is made. The procedure does not apply to certain issues: revenue, customs or administrative matters, state liability for acts and omissions in the exercise of state authority, matrimonial property regimes, bankruptcy, proceedings relating to the winding-up of insolvent companies or other legal persons, and judicial arrangements, social security, claims arising from non-contractual obligations, unless there was an agreement between the parties or an admission of debt or they relate to liquidated debts arising from joint ownership of property.

Generated:

a European order for payment procedure is established in the EU country where the claimant lives. Its purpose is to ensure that creditors and debtors have equal access to justice throughout the EU. The regulation also establishes an electronic system for determining which courts have jurisdiction to issue an order for payment, as well as a mechanism for the recovery of uncontested pecuniary claims.

Fig. 3 shows the distribution of Rouge 1 F1 scores for all experimented models. The PreSumm abstractive model curve is most similar to normal distribution. We can also observe similar behaviour between baseline extractive curve and PreSumm extractive. The rest have similar shapes and the fine-tuned T5 model achieves the best overall results.

5 Conclusion

We introduced a new dataset about summarization of European legislation documents. We also presented a comparative study of various algorithms for automatic text summarization on this dataset. In our experiments on these tasks, we obtained promising results including huge improvements over baseline. We believe that the new dataset, adopting existing algorithms to domain specific data and the results described in this paper will accelerate research directions on text summarization to expand the variety of domains with domain specific information and different sizes.

Acknowledgments

This research is partially supported by Project UNITe BG05M2OP001-1.001-0004 funded by the OP "Science and Education for Smart Growth" and
co-funded by the EU through the ESI Funds.

References


Not All Comments are Equal: Insights into Comment Moderation from a Topic-Aware Model

Elaine Zosa
University of Helsinki
elaine.zosa@helsinki.fi

Ravi Shekhar
Queen Mary University of London
r.shekhar@qmul.ac.uk

Mladen Karan
Queen Mary University of London
m.karan@qmul.ac.uk

Matthew Purver
Jožef Stefan Institute
m.purver@qmul.ac.uk

Abstract

Moderation of reader comments is a significant problem for online news platforms. Here, we experiment with models for automatic moderation, using a dataset of comments from a popular Croatian newspaper. Our analysis shows that while comments that violate the moderation rules mostly share common linguistic and thematic features, their content varies across the different sections of the newspaper. We therefore make our models topic-aware, incorporating semantic features from a topic model into the classification decision. Our results show that topic information improves the performance of the model, increases its confidence in correct outputs, and helps us understand the model’s outputs.

1 Introduction

Most newspapers publish their articles online, and allow readers to comment on those articles. This can increase user engagement and page views, and provides readers with an important route to public freedom of expression and opinion, with the ability to interact and discuss with others. Comment sections usually provide some degree of anonymity; while improving accessibility, this can also encourage inappropriate behaviour, and publishers therefore usually employ some moderation policy to regulate content and to ensure legal compliance (in some cases, publishers can be held responsible for user-contributed content on their sites).

One possible approach is a ‘moderate then publish’ policy, in which comments must be approved by a moderator before they appear; this requires significant manpower and introduces delays and limitations into the user conversation (for example, the New York Times only allows comments for one day after article publication2). On the other hand, a ‘publish then moderate’ strategy, in which comments are published immediately, and later removed if necessary, is less effective at blocking toxic or illegal content. Combined with the increase in comment volumes in recent years there is increasing interest in automatic moderation methods (see e.g. Pavlopoulos et al., 2017a), either as standalone tools or for integration into human moderators’ practices (Schabus and Skowron, 2018).

Detecting comments that need moderators’ attention is usually approached as a text classification task (see e.g. Pavlopoulos et al., 2017a); but comments can be blocked for a range of reasons (Shekhar et al., 2020). One is the presence of offensive language, a well-studied NLP task (see Section 2 below); however, others include advertising or spam, illegal content, spreading misinformation, trolling and incitement — all distinct categories which might be expected to show distinct features, and perhaps to vary according to the content being commented on. Another aspect that distinguishes the comment moderation task from the usual text classification tasks in NLP is the need for interpretable or explainable models: if classifiers are to be used by human moderators within publishers’ working practices, they must be able to understand the outputs (Švec et al., 2018).

Here, we therefore investigate models which can provide both an aspect of interpretability and the ability to take account of the topics being discussed, by incorporating topic information into the comment classifier. Specifically, we incorporate semantic representations learned by the Embedded Topic Model (ETM) (Dieng et al., 2020) into a classifier pipeline based on Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997). Our model improves performance

__________________________
1Some newspapers allow completely anonymous posting; some require commenters to create an account with a username, but this does not usually reveal their true identity.

2NYT Comment FAQ: https://nyti.ms/2PF02kj
by 4.4% over a text-only approach on the same dataset (Shekhar et al., 2020), and is more confident in the correct decisions it makes. Inspection of the topic distributions reveals how different newspaper sections have different language and topic distributions, including differences in the kind of comments that need moderation.3

2 Related Work

Automated news comment moderation  Most research on this task so far formulates it as a text classification problem: for a given comment, the model must predict whether the comment violates the newspaper’s policy. However, approaches to classification vary. Nobata et al. (2016) use a range of linguistic features, e.g. lexicon and n-grams. Pavlopooulos et al. (2017a) and Švec et al. (2018) use neural networks, specifically RNNs with an attention mechanism. Recently, Tan et al. (2020) and Tran et al. (2020) apply a modified BERT model (Devlin et al., 2019) while Schabus et al. (2017) use a bag-of-words approach.

Some approaches go beyond the comment text itself: Gao and Huang (2017) add information like user ID and article headline into their RNN to make the model context-aware; Pavlopooulos et al. (2017b) incorporate user embeddings; Schabus and Skowron (2018) incorporate the news category metadata of the article. However, no work so far investigates automatic modelling of topics (rather than relying on categorical metadata), or applies this to the comments rather than just their parent articles.

Some steps towards model interpretability and output explanation have also been taken: both Švec et al. (2018) and Pavlopooulos et al. (2017a) use an attention saliency map to highlight possibly problematic words. However, we are not aware of any work using higher-level topic information as a route to understanding model outputs.

Available datasets  Several datasets have been created for the news comment moderation task. Nobata et al. (2016) provide 1.43M comments posted on Yahoo! Finance and News over 1.5 years, in which 7% of the comments are labelled as abusive via a community moderation process. Gao and Huang (2017) contains 1.5k comments from Fox News, annotated with specific hateful/non-hateful labels as a post-hoc task, and having 28% hateful comments. However, both are relatively small, and their labelling methods mean that neither dataset is entirely representative of the moderation process performed by newspapers.

Pavlopooulos et al. (2017a) provides 1.6M comments from Gazzetta, a Greek sports news portal, over c.1.5 years. Here, 34% of comments are labelled as blocked, and the labels are derived from the newspaper’s human moderators and journalists. Schabus et al. (2017) and Schabus and Skowron (2018) provide a dataset from a German-language Austrian newspaper with 1M comments posted over 1 year, out of which 11,773 comments are annotated using seven different rules.

More recently, Shekhar et al. (2020) present a dataset from 24sata, Croatia’s most widely read newspaper. This dataset is significantly larger (10 years, c.20M comments); and moderator labels include not only a label for blocked comments, but also a record of the reason for the decision according to a 9-class moderation policy. However, their experiments show that classifier performance is limited, and transfers poorly across years. Here, we therefore use this dataset (see Section 3), with a view to improving performance and applying a topic-aware model to improve and better understand the robustness in the face of changing topics.

Related tasks  More attention has been given to related tasks, most prominently the detection of offensive language, hate speech, and toxicity (Pelicon et al., 2021). A comprehensive survey of dataset collection is provided by Poletto et al. (2020) and Vidgen and Derczynski (2020).5

Topic Modelling  Topic models capture the latent themes (also known as topics) from a collection of documents through the co-occurrence statistics of the words used in a document. Latent Dirichlet Allocation (LDA) (Blei et al., 2003), a popular method for capturing these topics, is a generative document model where a document is a mixture of topics expressed as a probability distribution over the topics and a topic is a distribution over the words in a vocabulary. The Embedded Topic Model (ETM, Dieng et al., 2020) is an LDA-like topic modelling method that exploits the semantic information captured in word embeddings during topic inference. The advantage of ETM over LDA

3Source code available at https://github.com/ezosa/topic-aware-moderation

4http://24sata.hr/

5http://hatespeechdata.com/ provides a comprehensive list of relevant datasets.
is that it combines the advantages of word embeddings with the document-level dependencies captured by topic modelling and has been shown to produce more coherent topics than regular LDA.

3 Dataset

We use the 24sata comment dataset (Shekhar et al., 2020; Pollak et al., 2021), introduced in Section 2. This contains c.21M comments on 476K articles from the years 2007-2019, written in Croatian. The dataset has details of comments blocked by the 24sata moderators, based on a set of moderation rules—these vary from hate speech to abuse to spam (see Shekhar et al., 2020, for rule description). The dataset also identifies the article under which a comment was posted, together with the section/sub-section of the newspaper the article appeared in. These sections/sub-sections relate to the content of the article: for example, the Sport section contains sports-related news while the Kolumne (Columns) section contains opinion pieces. The largest section, Vijesti (News), is further subdivided as shown in Table 2.

3.1 Data Selection

In this work, we use data from 2018 for training and validation of the topic model and classifiers and data from 2019 for testing. This reflects the realistic scenario where we use data collected from the past to make predictions. For training and validation, we randomly select 50,000 articles out of 65,989 articles from 2018, sampling from the nine most-representative sections/sub-sections (Table 2). Each article comes with c.50 comments on average.

To train the topic model, we sample around 80,000 comments across these articles, with a roughly equal split between blocked and non-blocked comments. This is to encourage a diverse mix of topics from both comment classes. As a preprocessing step we remove comments with less than 10 words from the training data (see Table 1 (lower part)). To train the classifiers, we randomly sample around 80,000 comments such that the sampled set has the same blocking rate as the entire 2018 dataset.

For the test set, we then use all 475,413 comments associated with the 17,953 articles from 2019. Table 1 (upper part) provides the dataset details, with comment moderation blocking rate. For the test set, Table 2 provides details on the section and sub-section of the related articles. These top nine sections account for more than 95% of the comments of the entire test set.

3.2 Content Analysis

To gain some insight into the content of blocked comments, we analyze the linguistic differences between blocked and non-blocked comments and across different sections. First, we compare comment length. As we can see from Table 3, blocked and non-blocked comments have, on average, similar lengths. However, if we further divide blocked comments into two sub-groups — spam and non-spam — we find that on average, spam comments are longer than other comments. We observe a similar pattern across different sections.

Next, we measure lexical diversity using mean-segmental type-token ratio (MSTTR). The MSTTR is computed as the mean of type-token ratio for every 1000 tokens in a dataset to control for dataset size (van Miltenburg et al., 2018). From Table 3, we see that non-blocked comments have higher MSTTR (i.e. higher lexical diversity) than blocked comments (0.62 vs 0.46). However, when we again divide blocked comments into spam and non-spam,
we observe that non-spam blocked comments have a similar MSTTR to non-blocked comments (0.61 vs 0.62), while spam comments have much lower MSTTR (0.35 vs 0.61). This suggests that blocked comments (excluding spam) have as rich a vocabulary as non-blocked. Again, we see a similar pattern across different news sections.

<table>
<thead>
<tr>
<th></th>
<th>Mean length</th>
<th>MSTTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>23.06</td>
<td>0.61</td>
</tr>
<tr>
<td>Non-blocked</td>
<td>23.01</td>
<td>0.62</td>
</tr>
<tr>
<td>Blocked</td>
<td>23.65</td>
<td>0.46</td>
</tr>
<tr>
<td>Blocked (non-spam)</td>
<td>19.16</td>
<td>0.61</td>
</tr>
<tr>
<td>Blocked (Spam only)</td>
<td>28.23</td>
<td>0.35</td>
</tr>
</tbody>
</table>

Table 3: Mean-segmental TTR and average length of comments

Now we analyse the usage of topics in our test set. For analysis, we extract the document-topic distribution of a comment in the set and rank the topics according to their weight in this mean distribution. We then take the top 15 topics for analysis because this is the average number of topics in a comment with a non-zero probability in our test set. Note that in this analysis we only use the document-topic

4.1 Topic Model

We use the Embedded Topic Model (ETM, Dieng et al., 2020) as our topic model since it has been shown to outperform regular LDA and other neural topic modelling methods such as NVDM (Miao et al., 2016). We also want to take advantage of ETM’s ability to incorporate the information encoded in pretrained word embeddings trained on vast amounts of data to produce more coherent topics. In the ETM, the topic-term distribution for topic $k$, $\beta_k$, is induced by a matrix of word embeddings $\rho$ and its respective topic embedding $\alpha_k$ which is a point in the word embedding space:

$$\beta_k = \text{softmax}(\rho^T \alpha_k)$$ (1)

The topic embeddings are learned during topic inference while the word embeddings can be pretrained or also learned during topic inference. In this work, we use pretrained embeddings.

The document-topic distribution of a document $d$, $\theta_d$, is drawn from the logistic normal distribution whose mean and variance come from an inference network:

$$\theta_d \sim LN(\mu_d, \sigma_d)$$ (2)

Given a trained ETM, we can infer the document-topic distribution (DTD) of an unseen document. In addition, we can also compute a document-topic embedding (DTE) as the weighted sum of the embeddings of the topics in a document, where the weight corresponds to the probability of the topic in that document:

$$DTE = \sum_{k=0}^{K} \alpha_k \theta_{d,k}$$ (3)

where $\alpha_k$ is the topic embedding of topic $k$, and $\theta_{d,k}$ is the probability of topic $k$ in doc $d$.

4.2 Topic Analysis

Now we analyse the usage of topics in our test set. We trained the ETM for 100 topics on the training set and inferred the topic distributions of the comments in the test set. For analysis, we extract the top topics in a set of comments. To do this, we take the mean of the topic distributions over the comments in the set and rank the topics according to their weight in this mean distribution. We then take the top 15 topics for analysis because this is the average number of topics in a comment with a non-zero probability in our test set. Note that in this analysis we only use the document-topic
distributions and not the document-topic embeddings. To more easily discuss the topics here we provide concise labels for each topic as interpreted by a native speaker. Automatic labelling of topics is a non-trivial task and an area of active research (Bhatia et al., 2016; Alokaili et al., 2020; Popa and Rebedea, 2021).

First, we examine the prevalent topics in the blocked and non-blocked comments, separately. The top topics of non-blocked comments cover a diverse range of subjects from politics to football while the top topics in blocked comments are dominated by spam and offensive language (Figure 1). However, we also see many topics shared between blocked and non-blocked comments.  

7 All 100 topics and labels are available at https://github.com/ezosa/topic-aware-moderation

Next we illustrate how different topics intersect and diverge between blocked and non-blocked comments across sections by looking at the top topics of two thematically-different sections, Lifestyle and Politika (Politics).

Figure 2 shows the top topics of these sections and the intersections between them. In Politics, blocked comments tend toward spam and targeted insults. Non-blocked topics include public safety and finances. However, we also see that more than half of the top topics overlap between blocked and non-blocked. This suggests that, thematically, there isn’t a very clear distinction between blocked and non-blocked comments in the Politics section.

In Lifestyle, blocked topics are dominated by spam and while there are topics on offensive insults, they are not as prevalent as the spam-related ones. The non-blocked topics are about family and relationships and commenters arguing with each other. Compared to Politics, we see a clearer distinction between topics in blocked and non-blocked in this section. In terms of topic overlaps between Lifestyle and Politics, blocked comments in both sections are dedicated to spam and insults while non-blocked comments focus on positive sentiments.

The combination of certain topics also provide an indication of the classification of the comment. For instance, we notice the use of topics about football cards in comments that do not do not discuss the sport (for instance, football cards as a topic is prominent in the blocked Lifestyle comments). It turns out that some commenters use the red and yellow cards from football as metaphors for being banned or having their comments blocked by moderators (12% of comments that use these metaphors are blocked by moderators). On the other hand, comments that use the football cards topics and any of the sports-related topics are likely to be a genuine discussion of football (only 5% of such comments are blocked by moderators). We show some examples of these comments in Table 5.

So clearly there is a distinction between the usage of topics in the non-blocked and blocked comments. We therefore think it is a good idea to propose a model which incorporates topic information into a comment moderation classifier.

5 Topic-aware Classifier

Our aim is to improve comment moderation predictions by combining textual features with document-level semantic information in the form of topics. To this end, we test several model architectures that combine a language model with topic features.

For the comment text representation, we use a...
bidirectional LSTM (BiLSTM, Schuster and Paliwal, 1997). The comment text is given as input to an embedding layer then a BiLSTM layer where the output of the final hidden state is taken as the encoded representation of the comment. For the topic representations, we use the topic distributions (DTD) and topic embeddings (DTE) discussed in Section 4.1.

We propose two fusion mechanisms to combine the text and topic representations: early and late fusion. In early fusion, topic features are concatenated with the output of the embedding layer and then passed to the BiLSTM layer. In EarlyFusion1 (EF1), only DTD is concatenated with the word embeddings; EarlyFusion2 (EF2) uses DTE instead of DTD; and EarlyFusion3 (EF3) uses both DTE and DTD. In late fusion, topic features are concatenated with the output representation of the BiLSTM layer, and passed to the MLP for classification. Again, LateFusion1 (LF1) uses DTD; LateFusion2 (LF2) uses DTE; and LateFusion3 (LF3) uses both. Figure 3 shows the architectures.

Our model is inspired by the Topic Compositional Neural Language Model (TCNLM, Wang et al., 2018) and the Neural Composite Language Model (NCLM, Chaudhary et al., 2020) that incorporate latent document-topic distributions with language models. Both of these models simultaneously learn a topic model and a language model through a joint training approach. The NCLM introduced the use of word embeddings to generate an explanatory topic representation for a document in addition to the document-topic proportions to produce the document-topic embeddings (DTE). Also unlike the TCNLM and NCLM, we use pre-trained topics in our model so as to easily de-couple and analyse the influence of topics in the classifier performance. Another related work is TopicRNN (Dieng et al., 2016), a model that uses topic proportions to re-score the words generated by the language model. The topics generated by this model, however, have been shown to have lower coherences compared to NCLM (Chaudhary et al., 2020).

6 Experimental Setup

Dataset As discussed in Section 3.1, we use the 2018 data as the training and validation sets of our topic-aware classifier and the 2019 data as the test set. Details of the train and validation sets are shown in Table 1 and the test set in Table 2.

Baseline models To assess how topic information improves comment classification, we use as baselines the following models trained only on text or topics:

- **Text only**: a classifier with BiLSTM & MLP layers, similar to Figure 3 but with comment text alone as input.
- **Document-topic distribution (DTD)**: MLP only, document-topic distributions as input.
- **Document-topic embedding (DTE)**: MLP only, document-topic embeddings as input.
- **DTD+E**: MLP only, concatenated document-topic distributions and embeddings.

Hyperparameters We use 300D word2vec embeddings, pretrained on the Croatian Web Corpus (HrWAC, Ljubešić and Erjavec, 2011; Šnajder, 2014), for training the ETM and to initialize the embedding layer of the BiLSTM. The ETM is trained...
for 500 epochs for 100 topics using the default hyperparameters from the original implementation. The BiLSTM is composed of one hidden layer of size 128 with dropout set to 0.5. The MLP classifier is composed of one fully-connected layer, one hidden layer of size 64, a ReLU activation, and a sigmoid for classification with the classification threshold set to 0.5. We use Adam optimizer with $lr = 0.005$. We train all classifiers for 20 epochs with early stopping based on the validation loss.

7 Results

In Table 4, we present the performance of the baselines and proposed models, measured as macro F1-scores. All models that combine text and topic representations perform better than the models that use only text or topics. Of the baseline models, the DTD model performs comparatively better than the DTE and DTD+E models, and surprisingly performs almost as well as the Text-only model; however, we show in Section 8 below that DTD is much less confident in its predictions than the Text-only model. Overall, the best performing model is LF1, which improves the Text-only model’s performance by +4.4% (67.37% vs 62.97%); and improves by a similar amount over Shekhar et al.’s results using mBERT (macro-F1 score 62.07 for year 2019).

Interestingly, we see a wide variation in performance across news sections. We observe that comments in Lifestyle and Tech are the easiest to classify (best F1 over 72.00) while Politika (Politics) is the most difficult (best F1 around 61.61). The main cause appears to be that Lifestyle and Tech have the highest proportion of spam comments: on average, 49.44% of blocked comments in the test set are spam, but for Lifestyle and Tech this number rises to 77.25% and 69.63%, respectively. As for the Politics section, the most likely reason the comments are difficult to classify is that, excluding spam, there is a high degree of overlap in the subjects discussed in the blocked and non-blocked comments (see the topic analysis in Section 4.2).

7.1 Analysis of Classifier Outputs

In general, we observe that blocked comments tend to use similar topics across different sections while non-blocked comments have more diverse topics. Of the nine sections that we analyzed, there are five topics that are prominent in blocked comments in all sections (‘Targeted/personal insults’, ‘Spam4’, ‘Spam7’, ‘Online media’, and, ‘Having a discussion’) and only three topics prominent in non-blocked comments (‘Having a discussion’, ‘Online media’, and, ‘Life and government’). This suggests that blocked comments are more semantically-coherent across sections than non-blocked ones. In contrast, topics in non-blocked comments tend to be more relevant to their respective sections: for instance, family and relationships are not discussed a lot in the Politics section, while Lifestyle commenters do not tend to talk about political issues.

The higher topical coherence then of blocked comments explains why a text classification approach can achieve reasonable performance; but the variation in blocked comment content between some sections explains why adding topic information improves our classification results.

Next, we analyze the confidence of classifiers and examine some of the outputs of the models. To analyze confidence, we gradually increase the classification threshold from 0.5 to 1.0 in increments of 0.05. For every new threshold, we plot the macro-F1 for the different models (Figure 4). We compare the confidence of four models: DTD, Text-only, EF2 (the strongest early fusion model), and LF1 (the overall best-performing model). We find that the most confident model is LF1 and the least confident is DTD. The two fusion classifiers display similar levels of confidence. The Text-only classifier is not as confident as the fusion classifiers but still more confident than DTD. This suggests that adding topic features to text not only improves performance, it also increases classifier confidence.

Figure 4: Confidence of the top performing models.

In Table 5 we give some examples of comments and the classifier decisions of the Text-only classifier and LF1 (our best-performing fusion model) and their top topics (topics with prob > 0.10). The
Table 4: Classifier performance measured as macro-F1.

<table>
<thead>
<tr>
<th>Comment</th>
<th>Label</th>
<th>Text-only</th>
<th>LF1</th>
<th>Top topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. konačno. gamad lopovska crno bijela prevarantska (finally. the black and white cheating thieving bastards)</td>
<td>1</td>
<td>1 (0.501)</td>
<td>1 (0.687)</td>
<td>Arguing a point, Political parties (offensive)</td>
</tr>
<tr>
<td>2. ...dobro jutro, moze crveni karton za novinara koji je osmislio naslov ;-) (... good morning, how about a red card for the journalist who came up with this title ;-)</td>
<td>1</td>
<td>0 (0.315)</td>
<td>0 (0.456)</td>
<td>Football cards</td>
</tr>
<tr>
<td>3. Ne bum komentiral, dosta mi je kazni od žutih i crvenih kartona. Strah me je cenzure i bradate cure. (No comment, I’m tired of getting yellow and red cards. I’m afraid of censorship and bearded ladies.)</td>
<td>0</td>
<td>0 (0.054)</td>
<td>0 (0.335)</td>
<td>Football cards, Random</td>
</tr>
<tr>
<td>4. Koji kurac Rumunjski sudac ne da koji karton više! Ce-hima. Pa svake tri minute sa leđa sruše Olma !!! (Why the fuck does the Romanian referee not give a few cards more to the Czechs, They tackle Olm from behind every three minutes.)</td>
<td>0</td>
<td>0 (0.303)</td>
<td>1 (0.587)</td>
<td>Targeted/personal insults</td>
</tr>
<tr>
<td>5. Baš ste jadnici kao i ovi sa 24sata koji u ovome uživaju! (All of you are lame as well as those from 24sata who enjoy this.)</td>
<td>1</td>
<td>0 (0.171)</td>
<td>0 (0.229)</td>
<td>Online media, Moderately offensive</td>
</tr>
<tr>
<td>6. Google sada plaća između 15.000 i 30.000 dolara mjesечно za rad na mreži od kuće. Pridružio sam se ovom poslu prije 3 mjeseci i zaradio 24857 dolara u prvom mjesecu ovog posla. &gt;&gt;&gt; URL (Google now pays between 15,000 and 30.000 dollars per month for working remotely from home. I started this job 3 months ago and made 24857 dollars in the first month of this job. &gt;&gt;&gt; URL)</td>
<td>0</td>
<td>1 (0.67)</td>
<td>1 (0.90)</td>
<td>Spam4</td>
</tr>
</tbody>
</table>

Table 5: Sample comments and classifier decisions.

first example contains swearing which both models pick up on and classify as blocked although LF1 is more confident in its decision than Text-only. In the second example, both models predict the wrong label but LF1 treats this as a borderline case because it is targeted at the moderators. However since this is only a mild provocation of the moderators, this might be a case where the gold label is incorrect. The topics also pick up on the fact that this comment talks about football cards but only has a tenuous connection to the sport (“getting a red card” is an expression used for “being banned”). In contrast, the third comment also uses the banning sense of “card” but is not directed at anyone, and is thus labeled as 0 (non-blocked), which both models get right. Again the topics indicate that the comment is not really about the sport. The fourth example shows a case where “cards” are mentioned in their standard football sense but also contains a swear word, making the gold label of 0 (non-blocked) questionable. The better performance of LF1 on such examples, compared to Text-only, implies that
LF1 is better aware of the different semantics of “card” (sports-related vs. metaphorical), likely due to added topic information.

The fifth example contains a moderately offensive insult that is not directed at any single group except the 24sata readership in general. One reason why both classifiers do not get this right is that the word jadnici is not strong enough to be considered offensive. Finally the last example is clearly a spam comment that both classifiers correctly classify but for which the gold label is incorrect.

Overall, compared to the Text-only model, we find that LF1 more often than not improves the confidences (and sometimes the classification), especially in cases in which the gold label is clear. This is valuable in practice, as better confidences might lead to better prioritisation of comments for manual moderation, reducing the time required to remove the most problematic ones.

8 Conclusion

In this work, we propose a model to combine document-level semantics in the form of topics with text for comment moderation. Our analysis shows that blocked and non-blocked comments have different linguistic and thematic features, and that topics and language use vary considerably across news sections, including some variation in the comments that should be blocked. We also found that blocked comments tend to be more semantically coherent across sections than non-blocked ones. We therefore see that the use of topics in our model improves performance, and gives more confident outputs, over a model that only uses the comment text. The model also provides topic distributions, interpretable as keywords, as a form of an explanation of its prediction. As future work, we plan to incorporate comment, article, and user metadata into the model.

Acknowledgements

This work has been supported by the European Union Horizon 2020 research and innovation programme under grants 770299 (NewsEye) and 825153 (EMBEDDIA), and by the UK EPSRC under grant EP/S033564/1.

Ethics and Impact Statement

Data The dataset and annotations are provided by the publisher of 24sata.hr, Styria Media Group, for research purposes and deposited in the CLARIN repository. The authors of the comments are anonymised. The researchers used the data as-is and did not modify or add annotations.

Intended Use The models we present here are intended to assist comment moderators in their work. We do not recommend that the model be deployed in the moderation process without a human-in-the-loop.

Potential Misuse The models and the analysis of their performance we provide in this paper could be used by malicious actors to gain an insight into the comment moderation process and find loopholes in the process. However, we think such a risk is unlikely and the impact it might have outweighs the potential benefits of models intended to assist human moderators such as the ones we present here.

References


Author Index

A. Soares, Tayane, 239
Abdi Ghavidel, Hadi, 306
Abdibayev, Almas, 1
Aceta, Cristina, 10
Aggarwal, Charu C., 571
Aggarwal, Salil, 19, 732
Ahmad, Zishan, 1134
Ahmadnia, Benyamin, 26
Al-Hajj, Moustafa, 40
Al Sharou, Khetam, 58
Ala, Hema, 31
Alam, Firoj, 1001, 1014
Almeida Cruz, Yudivián, 221, 407
Alotaibi, Naif, 49
Amin, Mohammad Ruhul, 468
Anchíeta, Rafael, 1265
André Gonçalves, Marcos, 239
Anggraito, Adityo, 84
Aoki, Tatsuya, 1610
Araújo de Britto, Felipe, 68
Argese, Chiara, 1530
Arçan, Bilge Nas, 766
Armengol-Estapé, Jordi, 78
Artari, Valentina Kania Prameswara, 84
Artemova, Ekaterina, 880
Atsumi, Masayasu, 993
Auersperger, Michal, 91
Avgustinova, Tania, 279, 972
Avram, Andrei-Marius, 97
Ayetiran, Eniafe Festus, 1072
Bačkovský, Dalibor, 1072
Bajaj, Vaibhav, 107
Ballal, Husamelddin, 116
Basile, Angelo, 133
Basile, Valerio, 126
Bastos Fóscolo, Rodrigo, 239
Battistelli, Delphine, 954
Batura, Tatiana, 880
Bear, Diego, 143
Béchet, Nicolas, 954
Belemkoabga, David Stéphane, 152
Benedetto, Luca, 851
Benevenuto, Fabrício, 1442
Benoit, Dries, 851
Benyekhlef, Karim, 1238
Berlanga Neto, Paulo, 160
Bijoy, Biddut Sarker, 468
Blanco, Alberto, 170
Blin, Kevin, 178
Borin, Lars, 1475
Bossard, Aurélien, 152
Bouillon, Pierre-ette, 1054
Boumber, Dainis, 552
Bout, Andrey, 213
Boytcheva, Svetla, 562, 1452
Bozhanova, Krasimira, 187, 1384
Brandt Skelbye, Molly, 195
Braslavski, Pavel, 880
Brkic, Marija, 1113
Broni-Bediako, Clifford, 993
Bucur, Ana-Maria, 204
Budi, Indra, 84
Burnyshev, Pavel, 213
Byrd, Andrew Miles, 1459
Cabrio, Elena, 1502
Cagnazzo, Christian, 126
Cañizares-Díaz, Hian, 221
Caragea, Cornelia, 1620
Carlson, Keith, 231
Carvalho, Isabel, 1442
Casillas, Arantza, 170
Castaldo, Maria, 187
Castro Ferreira, Thiago, 68, 239
Cattan, Oralie, 249
Černiauskas, Algimantas, 1494
Cesur, Neslihan, 766
Chaluvadi, Anudeep, 749
Charnois, Thierry, 861
Chathuranga, Shanaka, 261
Chatzipanagiotou, Marita, 713
Chay-intr, Thodsaporn, 269
Chechev, Milen, 1649
Chen, Yu, 279
Cho, Myong-ho, 689
Cho, Song-yong, 689
Choi, Hee-Soo, 286
<table>
<thead>
<tr>
<th>Name</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choo, Kenny</td>
<td>1640</td>
</tr>
<tr>
<td>Ciobotaru, Alexandra</td>
<td>296</td>
</tr>
<tr>
<td>Çöltekin, Çağrı</td>
<td>899</td>
</tr>
<tr>
<td>Cook, Paul</td>
<td>143, 697</td>
</tr>
<tr>
<td>Corbeil, Jean-Philippe</td>
<td>306</td>
</tr>
<tr>
<td>Cortis, Keith</td>
<td>314</td>
</tr>
<tr>
<td>Costa-jussà, Marta R.</td>
<td>78</td>
</tr>
<tr>
<td>Cremonesi, Paolo</td>
<td>851</td>
</tr>
<tr>
<td>Cristea, Alina Maria</td>
<td>320</td>
</tr>
<tr>
<td>Cumbicus-Pineda, Oscar M.</td>
<td>329</td>
</tr>
<tr>
<td>Cunha, Rossana</td>
<td>239</td>
</tr>
<tr>
<td>Da San Martino, Giovanni</td>
<td>1001, 1014, 1601</td>
</tr>
<tr>
<td>Dalianis, Hercules</td>
<td>170, 795, 1162</td>
</tr>
<tr>
<td>Dannéls, Dana</td>
<td>195, 1467, 1475</td>
</tr>
<tr>
<td>Danner, Hannah</td>
<td>480</td>
</tr>
<tr>
<td>Dascalu, Mihai</td>
<td>433</td>
</tr>
<tr>
<td>Davis, Brian</td>
<td>314</td>
</tr>
<tr>
<td>de Chalendar, Gaël</td>
<td>861</td>
</tr>
<tr>
<td>De Clercq, Orphee</td>
<td>351</td>
</tr>
<tr>
<td>de Melo, Gerard</td>
<td>519</td>
</tr>
<tr>
<td>Delany, Sarah Jane</td>
<td>116</td>
</tr>
<tr>
<td>Denisov, Iliia</td>
<td>880</td>
</tr>
<tr>
<td>Deschamps, Arthur</td>
<td>340</td>
</tr>
<tr>
<td>Desot, Thierry</td>
<td>351</td>
</tr>
<tr>
<td>Dinkov, Yoan</td>
<td>187</td>
</tr>
<tr>
<td>Dinu, Anca</td>
<td>320, 363</td>
</tr>
<tr>
<td>Dinu, Liviu P.</td>
<td>204, 296, 320</td>
</tr>
<tr>
<td>Dorr, Bonnie</td>
<td>26</td>
</tr>
<tr>
<td>Dowlagar, Suman</td>
<td>372</td>
</tr>
<tr>
<td>Dragut, Eduard</td>
<td>552, 1427, 1557</td>
</tr>
<tr>
<td>Du, Mingqian</td>
<td>1540</td>
</tr>
<tr>
<td>Durandin, Oleg</td>
<td>669</td>
</tr>
<tr>
<td>Durgar El-Kahlout, Ilknur</td>
<td>1421</td>
</tr>
<tr>
<td>Dutta Chowdhury, Koel</td>
<td>380</td>
</tr>
<tr>
<td>Ebling, Sarah</td>
<td>1343</td>
</tr>
<tr>
<td>Ekbal, Asif</td>
<td>1134</td>
</tr>
<tr>
<td>Elfdaeel, Haytham</td>
<td>391</td>
</tr>
<tr>
<td>Emelyanov, Yaroslav</td>
<td>399</td>
</tr>
<tr>
<td>Escolano, Carlos</td>
<td>78</td>
</tr>
<tr>
<td>España-Bonet, Cristina</td>
<td>380</td>
</tr>
<tr>
<td>Essa, Abdallah</td>
<td>152</td>
</tr>
<tr>
<td>Estevanell-Valladares, Ernesto L.</td>
<td>407</td>
</tr>
<tr>
<td>Estevez-Velarde, Suilan</td>
<td>221, 407</td>
</tr>
<tr>
<td>Etchegoyhen, Thierry</td>
<td>416</td>
</tr>
<tr>
<td>F. Oliveira, Rivane</td>
<td>239</td>
</tr>
<tr>
<td>Feng, Yukun</td>
<td>426</td>
</tr>
<tr>
<td>Fernández, Izaskun</td>
<td>10</td>
</tr>
<tr>
<td>Fersini, Elisabetta</td>
<td>1412</td>
</tr>
<tr>
<td>Forsberg, Markus</td>
<td>1475</td>
</tr>
<tr>
<td>Fort, Karèn</td>
<td>286</td>
</tr>
<tr>
<td>Foster, Daniel</td>
<td>1484</td>
</tr>
<tr>
<td>França, Celso</td>
<td>239</td>
</tr>
<tr>
<td>Franco-Salvador, Marc</td>
<td>133</td>
</tr>
<tr>
<td>François, Thomas</td>
<td>1054, 1200</td>
</tr>
<tr>
<td>Frank, Victor M.</td>
<td>588</td>
</tr>
<tr>
<td>Frode de la Foret, Pierre</td>
<td>433</td>
</tr>
<tr>
<td>Frota, Gabriel</td>
<td>239</td>
</tr>
<tr>
<td>Gaikwad, Saurabh Sampatrao</td>
<td>442</td>
</tr>
<tr>
<td>Galitsky, Boris</td>
<td>449</td>
</tr>
<tr>
<td>García Flores, Jorge</td>
<td>861</td>
</tr>
<tr>
<td>Georgescu, Simona</td>
<td>320</td>
</tr>
<tr>
<td>Gerginov, Simeon</td>
<td>1452</td>
</tr>
<tr>
<td>Gete, Harrítxu</td>
<td>416</td>
</tr>
<tr>
<td>Ghassem-Sani, Gholamreza</td>
<td>652</td>
</tr>
<tr>
<td>Ghosh, Mainak</td>
<td>480</td>
</tr>
<tr>
<td>Glass, James</td>
<td>1601</td>
</tr>
<tr>
<td>Gnawali, Omprakash</td>
<td>1312, 1427</td>
</tr>
<tr>
<td>Goinđani, Akshay</td>
<td>459</td>
</tr>
<tr>
<td>Gollapalli, Sujatha D.</td>
<td>340</td>
</tr>
<tr>
<td>Goncharova, Elizaveta</td>
<td>449</td>
</tr>
<tr>
<td>Gonzalez-Dios, Itziar</td>
<td>329</td>
</tr>
<tr>
<td>Gorelick, Henry</td>
<td>468</td>
</tr>
<tr>
<td>Gravier, Guillaune</td>
<td>890</td>
</tr>
<tr>
<td>Groh, Georg</td>
<td>480, 1519</td>
</tr>
<tr>
<td>Guillaume, Bruno</td>
<td>286</td>
</tr>
<tr>
<td>Gutierrez, Yoan</td>
<td>407</td>
</tr>
<tr>
<td>Gutíérrez, Yoan</td>
<td>221</td>
</tr>
<tr>
<td>Hagerer, Gerhard</td>
<td>480, 1519</td>
</tr>
<tr>
<td>Hämäläinen, Mika</td>
<td>1530</td>
</tr>
<tr>
<td>Han, Yong-jun</td>
<td>689</td>
</tr>
<tr>
<td>Harrando, Ismail</td>
<td>488</td>
</tr>
<tr>
<td>Harrison, Brent</td>
<td>1459</td>
</tr>
<tr>
<td>Hasan Dalip, Daniel</td>
<td>239</td>
</tr>
<tr>
<td>Hatua, Amartya</td>
<td>499</td>
</tr>
<tr>
<td>Hauer, Bradley</td>
<td>509</td>
</tr>
<tr>
<td>He, Qing</td>
<td>1540</td>
</tr>
<tr>
<td>He, Xiaoli</td>
<td>519</td>
</tr>
<tr>
<td>Henriksson, Aron</td>
<td>795</td>
</tr>
<tr>
<td>Hercig, Tomáš</td>
<td>529</td>
</tr>
<tr>
<td>Hettiarachchi, Hansi</td>
<td>1436</td>
</tr>
<tr>
<td>Hiai, Satoshi</td>
<td>535</td>
</tr>
<tr>
<td>Himeno, Takumi</td>
<td>543</td>
</tr>
<tr>
<td>Homan, Christopher</td>
<td>442</td>
</tr>
<tr>
<td>Hosseinia, Marjan</td>
<td>552</td>
</tr>
<tr>
<td>Hoste, Veronique</td>
<td>351</td>
</tr>
<tr>
<td>Hristov, Anton</td>
<td>562</td>
</tr>
<tr>
<td>Hu, Chenlong</td>
<td>426, 1590</td>
</tr>
<tr>
<td>Hu, Zhiqiang</td>
<td>571</td>
</tr>
<tr>
<td>Hubková, Helena</td>
<td>581</td>
</tr>
<tr>
<td>Name</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Hull, James R.</td>
<td>588</td>
</tr>
<tr>
<td>Hwang, Hohyun</td>
<td>598</td>
</tr>
<tr>
<td>Idicula, Sumam Mary</td>
<td>1171</td>
</tr>
<tr>
<td>Igarashi, Yohei</td>
<td>1184</td>
</tr>
<tr>
<td>Ikonić Nešić, Milica</td>
<td>1256</td>
</tr>
<tr>
<td>Ilchev, Artur</td>
<td>606</td>
</tr>
<tr>
<td>Ilvovsky, Dmitry</td>
<td>449</td>
</tr>
<tr>
<td>Im, Ju-song</td>
<td>689</td>
</tr>
<tr>
<td>Imperial, Joseph Marvin</td>
<td>616</td>
</tr>
<tr>
<td>Inácio, Marcio</td>
<td>624</td>
</tr>
<tr>
<td>Indurkhya, Sagar</td>
<td>634</td>
</tr>
<tr>
<td>Islam, Md Saiful</td>
<td>468</td>
</tr>
<tr>
<td>Itoh, Youki</td>
<td>645</td>
</tr>
<tr>
<td>Ivanov, Ivan</td>
<td>1384</td>
</tr>
<tr>
<td>Ivanov, Ivaylo</td>
<td>1452</td>
</tr>
<tr>
<td>Ivanov, Philip</td>
<td>1452</td>
</tr>
<tr>
<td>Ivanov, Vladimir</td>
<td>880</td>
</tr>
<tr>
<td>Iwakura, Tomoya</td>
<td>535</td>
</tr>
<tr>
<td>Jain, Sakshi C.</td>
<td>1134</td>
</tr>
<tr>
<td>Jalali Farahani, Farane</td>
<td>652</td>
</tr>
<tr>
<td>Jannatus Saba, Syeda</td>
<td>468</td>
</tr>
<tr>
<td>Jarrar, Mustafa</td>
<td>40</td>
</tr>
<tr>
<td>Jiang, Jing</td>
<td>1391</td>
</tr>
<tr>
<td>Jiwanggi, Meganingrum Arista</td>
<td>84</td>
</tr>
<tr>
<td>Joy, Mike</td>
<td>49</td>
</tr>
<tr>
<td>Juola, Patrick</td>
<td>1178</td>
</tr>
<tr>
<td>Jurek-Loughrey, Anna</td>
<td>963</td>
</tr>
<tr>
<td>Kahla, Mram</td>
<td>660</td>
</tr>
<tr>
<td>Kakadiaris, Ioannis</td>
<td>1301</td>
</tr>
<tr>
<td>Kalycheva, Ekaterina</td>
<td>669</td>
</tr>
<tr>
<td>Kameswari, Lalitha</td>
<td>676</td>
</tr>
<tr>
<td>Kamigaito, Hidetaka</td>
<td>269</td>
</tr>
<tr>
<td>Kaneko, Kimi</td>
<td>1579</td>
</tr>
<tr>
<td>Kar, Sudipta</td>
<td>468</td>
</tr>
<tr>
<td>Karan, Mladen</td>
<td>1656</td>
</tr>
<tr>
<td>Kikutu, Naokyo</td>
<td>684</td>
</tr>
<tr>
<td>Kim, Kwang-hyok</td>
<td>689</td>
</tr>
<tr>
<td>King, Milton</td>
<td>697</td>
</tr>
<tr>
<td>Kirchhoff, Martin</td>
<td>480</td>
</tr>
<tr>
<td>Klakow, Dietrich</td>
<td>972</td>
</tr>
<tr>
<td>Kobayashi, Naoki</td>
<td>775</td>
</tr>
<tr>
<td>Kocarev, Ljupco</td>
<td>914</td>
</tr>
<tr>
<td>Koeva, Svetlana</td>
<td>706</td>
</tr>
<tr>
<td>Kondrak, Grzegorz</td>
<td>509</td>
</tr>
<tr>
<td>König, Alexandra</td>
<td>835</td>
</tr>
<tr>
<td>Konopík, Miloslav</td>
<td>1123</td>
</tr>
<tr>
<td>Korre, Katerina</td>
<td>713</td>
</tr>
<tr>
<td>Koychev, Ivan</td>
<td>187</td>
</tr>
<tr>
<td>Kral, Pavel</td>
<td>529</td>
</tr>
<tr>
<td>Kravchev, Boris</td>
<td>1452</td>
</tr>
<tr>
<td>Krüger, Insa</td>
<td>835</td>
</tr>
<tr>
<td>Krstev, Cvetana</td>
<td>1256</td>
</tr>
<tr>
<td>Kübler, Sandra</td>
<td>872</td>
</tr>
<tr>
<td>Kucharavy, Andrei</td>
<td>178</td>
</tr>
<tr>
<td>Kugathasan, Archchanan</td>
<td>723</td>
</tr>
<tr>
<td>Kumano, Yasutaka</td>
<td>1579</td>
</tr>
<tr>
<td>Kumar, Sourav</td>
<td>19</td>
</tr>
<tr>
<td>Kunilovskaya, Maria</td>
<td>739</td>
</tr>
<tr>
<td>Kusampudi, Siva Subrahmanan</td>
<td>749</td>
</tr>
<tr>
<td>Kuyrükçu, Oguzhan</td>
<td>766</td>
</tr>
<tr>
<td>Kuzgun, Asli</td>
<td>766</td>
</tr>
<tr>
<td>Kwon, Jingun</td>
<td>775</td>
</tr>
<tr>
<td>Labruna, Tiziano</td>
<td>785</td>
</tr>
<tr>
<td>Lam, Wai</td>
<td>1322</td>
</tr>
<tr>
<td>Lamproutsidis, Anastasios</td>
<td>795</td>
</tr>
<tr>
<td>Langbehn, Abisague</td>
<td>239</td>
</tr>
<tr>
<td>Langlais, Philippe</td>
<td>1238</td>
</tr>
<tr>
<td>Langlois, Quentin</td>
<td>1200</td>
</tr>
<tr>
<td>Lapshinova-Koltunski, Ekaterina</td>
<td>739</td>
</tr>
<tr>
<td>Lazarova, Gergana</td>
<td>1649</td>
</tr>
<tr>
<td>Le Nouy, Michiel</td>
<td>890</td>
</tr>
<tr>
<td>Lecorvé, Gwénoëlle</td>
<td>954</td>
</tr>
<tr>
<td>Lee, Isabela</td>
<td>239</td>
</tr>
<tr>
<td>Lee, John</td>
<td>803</td>
</tr>
<tr>
<td>Lee, Joosung</td>
<td>810</td>
</tr>
<tr>
<td>Lee, Roy Ka-Wei</td>
<td>571</td>
</tr>
<tr>
<td>Lee, Younghoon</td>
<td>598</td>
</tr>
<tr>
<td>Lejeune, Gaël</td>
<td>1231</td>
</tr>
<tr>
<td>Leonova, Viktorija</td>
<td>819</td>
</tr>
<tr>
<td>Li, Zhenhao</td>
<td>58</td>
</tr>
<tr>
<td>Li, Zhi</td>
<td>1640</td>
</tr>
<tr>
<td>Lin, Ruixi</td>
<td>829</td>
</tr>
<tr>
<td>Lindsay, Hali</td>
<td>835</td>
</tr>
<tr>
<td>Linz, Nicklas</td>
<td>835</td>
</tr>
<tr>
<td>Lisena, Pasquale</td>
<td>488</td>
</tr>
<tr>
<td>Liu, Ye</td>
<td>844</td>
</tr>
<tr>
<td>Ljubešić, Nikola</td>
<td>914</td>
</tr>
<tr>
<td>Llorens Salvador, Marisa</td>
<td>116</td>
</tr>
<tr>
<td>Loginova, Ekaterina</td>
<td>851</td>
</tr>
<tr>
<td>López Espejel, Jessica</td>
<td>861</td>
</tr>
<tr>
<td>Lopez Long, Holly</td>
<td>872</td>
</tr>
<tr>
<td>Loukachevitch, Natalia</td>
<td>880</td>
</tr>
<tr>
<td>Luan, Yixing</td>
<td>509</td>
</tr>
<tr>
<td>Luisa A. R. Guimarães, Ana</td>
<td>239</td>
</tr>
<tr>
<td>Lupták, Dávid</td>
<td>1072</td>
</tr>
<tr>
<td>Magniní, Bernardo</td>
<td>785</td>
</tr>
<tr>
<td>Mahendra, Rahmad</td>
<td>84</td>
</tr>
<tr>
<td>Maier, Wolfgang</td>
<td>844</td>
</tr>
<tr>
<td>Majumder, Navonil</td>
<td>1035</td>
</tr>
</tbody>
</table>
Rajitha, Charith, 1154
Ramakers, Inez HGB, 835
Ramnan, Roshni, 1134
Ran, Jinye, 1640
Ranasinghe, Tharindu, 442, 1436
Ranathunga, Surangika, 261, 1154
Remmer, Sonja, 170, 1162
Renjit, Sara, 1171
Riddell, Allen, 1, 231, 1178, 1184
Rigotto, Isabela, 239
Rios, Annette, 1343
Robertson, Frankie, 1192
Rockmore, Daniel, 1, 231
Rodrigues de Góes, Fabiana, 1442
Rodrigues, Christophe, 152
Rodrigues, Paul, 588
Rolim, Sophia, 239
Rolin, Eva, 1200
Rosset, Sophie, 249
Rosso, Paolo, 1248
Roy, Archishman, 480
Rozovskaya, Alla, 1210
Ruiz, Evandro Eduardo Seron, 160
Ruiz, Victor, 416
Ruseti, Stefan, 433
Ryang, Chol-ho, 689
Rytting, C. Anton, 588
Rzepka, Rafal, 1579
Saadany, Hadeel, 1221
Sachintha, Dilan, 1154
Sahnoun, Sihem, 1231
Salatín, Olivier, 1238
Saleem, Raheela, 1484
Sánchez-Junquera, Javier, 1248
Sandescu, Cristian, 433
Šandrih Todorović, Branislava, 1256
Sanıyar, Ezgi, 766
Santos, Ariel, 239
Saraiva, Ghivvago Damas, 1265
Sathineni, Preetham, 758
Saveleva, Ekaterina, 1272
Sazzed, Salim, 1285, 1293
Sébèlliot, Pascale, 890
Seják, Michal, 1330
Sengupta, Shubhashis, 1134
Servan, Christophe, 249
Shaar, Shaden, 1001, 1014
Shafaei, Mahsa, 1301
Shahriar, Sadat, 1312
Sharma, Dipti, 31, 732
Sheikh Muhammad, Azam, 1467, 1484
Sheinin, Vadim, 1494
Shekhar, Ravi, 1656
Shen, Xin, 1322
Shimada, Kazutaka, 535, 543, 1579
Shinnou, Hiroyuki, 645, 684
Shrivastava, Manish, 459
Sido, Jakub, 1123, 1330
Silva Parreiras, Fernando, 68
Simov, Kiril, 905
Slavcheva, Milena, 905
Smailis, Christos, 1301
Sojka, Petr, 1072
Solorio, Thamar, 1301
Soroa, Aitor, 10, 329
Sorokin, Nikita, 606
Specia, Lucia, 58
Spring, Nicolas, 1343
Stajner, Sanja, 1354, 1364
Stanković, Ranka, 1256
Štefánik, Michal, 1072
Steinberger, Josef, 1142
Stenger, Irina, 972
Stöckl, Andreas, 1373
Strzyz, Michalina, 982
Sukhareva, Maria, 1088
Sumathipala, Sagara, 723
Swahn, Matthew, 588
Świeczkowska, Patrycja, 1579
Sylla, Kévin, 152
Tagarev, Andrey, 1384
Tahchiev, Aleksandar, 562
Takamura, Hiroya, 426, 775, 1590, 1610
Tamura, Akihiro, 1571
Tan, Minghuan, 1391, 1401
Tantavy, Ashraf, 1221
Terragni, Silvia, 1412
Topçu, Berkay, 1421
Travadel, Sebastien, 433
Tröger, Johannes, 835
Troncy, Raphael, 488
Tsonkov, Todor, 1649
Tufis, Dan Ioan, 97
Tulechki, Nikola, 562
Tumarada, Kishore, 1427
Tutubalina, Elena, 880
Uban, Ana Sabina, 320
Ultes, Stefan, 844
Upadhayay, Ishan, 107
Ureña-López, L. Alfonso, 1100
Uyangodage, Lasitha, 1436
van Genabith, Josef, 380
Vandeghinste, Vincent, 1054
Vargas, Francielle, 1442
Vassileva, Sylvia, 1452
Velichkov, Boris, 1452
Venturini, Tommaso, 187
Verhey, Frans RJ, 835
Verma, Kanishk, 314
Verma, Rakesh, 499
Victor de Pinho Costa, João, 239
Vilares, David, 982
Villata, Serena, 1502
Vinogradov, Anton, 1459
Virk, Shafqat Mumtaz, 1467, 1475, 1484
Vo, Ngoc Phuoc An, 1494
Vorakitphan, Vorakit, 1502

Wang, Haining, 1178
Wang, Wenyong, 1540
Watanabe, Taiki, 535
Watrin, Patrick, 1200
Wawer, Aleksander, 1512
Way, Andy, 1113
Wich, Maximilian, 1519
Widmer, Christian, 1519
Wiechetek, Linda, 1530

Xie, Yikuan, 1540

Yanamandra, Venkata Himakar, 1549
Yang, Fan, 1427, 1557
Yang, Haoran, 1563
Yang, Zijian Gyöző, 660
Yano, Yuki, 1571
Yenice, Arife Betül, 766
Yenikent, Seren, 1364
Yeung, Chak Yan, 803
Yıldız, Oğuz Kerem, 766
Yıldız, Olcay Taner, 766
Yoshida, Hiroaki, 1579
Yoshikawa, Hiyori, 1579
You, Jingyi, 1590
Yu, Seunghak, 1601

Zampieri, Marcos, 442
Zdravevski, Eftim, 914
Zeghari, Radia, 835
Zhang, Yifan, 1001, 1014, 1427
Zhang, Ying, 1610
Zhao, Chenye, 1620
Zhao, Jiaxi, 480
Zhou, He, 1630
Zhu, Jiawen, 1640
Zmiycharov, Valentin, 1649
Zoicas, Laurentiu, 320
Zosa, Elaine, 1656
Zuters, Janis, 819